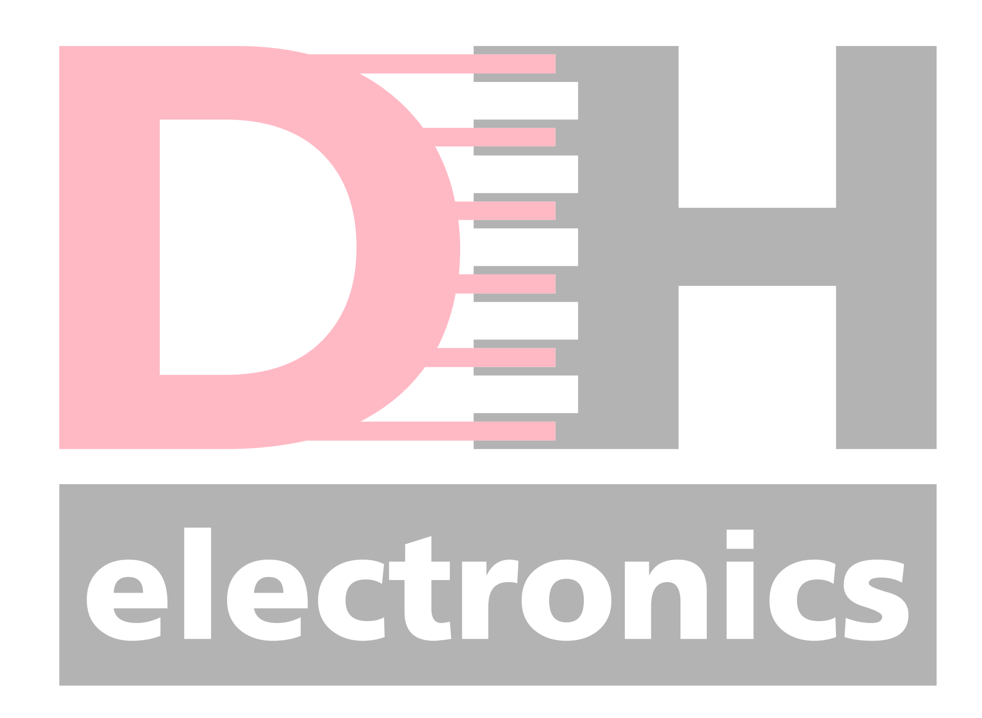
de

Bedienungsanleitung XLON® USB Adapter

Version 1.0



DH electronics GmbH

Am Anger 8
83346 Bergen
Germany
Tel.: +49 8662 4882-0
Fax.: +49 8662 4882-99
E-Mail: info@xlon.de
www.xlon.de





Bedienungsanleitung
XLON® USB Adapter

Version 1.0



de


XLON
USB

1 Zu diesem Handbuch

Dieses Handbuch soll den Anwender bei der Installation und Konfiguration des XLON USB Adapters unterstützen.

Dem Entwickler werden gleichzeitig Informationen zur Anbindung bzw. Erstellung geeigneter Anwendungssoftware gegeben.

Verwendete Piktogramme und Symbole

In dieser Anleitung werden folgende Piktogramme und Symbole verwendet, um auf besondere Punkte aufmerksam zu machen:



Achtung! Besonders wichtiger, sicherheitsrelevanter Punkt.



Verletzungsgefahr durch elektrische Spannung/Strom.



Gefahr der Beschädigung elektronischer Bauteile durch statische Aufladung.



Verletzungsgefahr durch mechanische Bauteile.

• (Boller) = Aufzählungszeichen, durchzuführende Tätigkeiten/Arbeitsschritte

☞ Hinweis! Besonders zu beachten.

2 Einleitung / Produktinformation

Der XLON USB Adapter ermöglicht den Anschluss eines PC oder Notebook an ein LonWorks® Netzwerk über den Universal Serial Bus, kurz USB. Er ist konzeptioniert für den Einsatz in der Industrie-, der Prozess- und der Gebäudeautomation.

Der XLON USB Adapter unterstützt sowohl das LNS Netzwerk-Service-Interface (NSI) für alle LNS-Tools, als auch das Microprozessor-Interface-Program (MIP) für selbst erstellte Anwendungen.

Das LNS-Netzwerk-Betriebssystem ermöglicht auf Grund seiner Client-Server-Architektur den gleichzeitigen Zugriff unterschiedlichster Anwendungen auf den Netzwerk-Service-Server (NSS). Dadurch können Tools unterschiedlichster Hersteller zur gleichen Zeit Installation, Wartung, Überwachung und Steuerung im LonWorks®-Netzwerk durchführen. Mit Hilfe des XLON USB Adapters ist es auch möglich, einen PC als äußerst leistungsfähigen LonWorks®-Knoten zu betreiben. Hierbei läuft auf dem PC die LonWorks®-Anwendung und der XLON USB Adapter ist für die Verarbeitung des LonTalk®-Protokolls zuständig. Im Vergleich zu einem Neuron® basierenden Knoten ermöglicht dies wesentlich höhere Rechenleistungen in einer LonWorks®-Anwendung. Zudem ist die Anzahl der möglichen Netzwerk-Variablen von 62 auf 4096 beträchtlich erweitert, was häufig für Wartungs- und Überwachungsanwendungen wichtig ist.

Der XLON USB Adapter hat entweder einen integrierten FTT-10A Transceiver für Free

Topology und Link Power Netzwerke, oder einen RS485 Transceiver für Twisted-Pair-Netzwerke.

Der XLON® USB Adapter besitzt zur Visualisierung der Betriebszustände eine Service- und eine Status-Leuchtdiode. Für die manuelle Installation ist ein Service Pin Taster nach außen geführt.
Die downloadbare Firmware ermöglicht Updates ohne Zugriff auf die Hardware.

Beispielprogramme für den Zugriff auf die Treiber unter C/C++ und VisualBasic zum Download finden Sie unter:

http://www.xlon.de

Lieferumfang

- XLON® USB Gerät
- USB Kabel (1 Meter)
- Weidmüller-Klemme für LonWorks®-Netzwerk-Anschluß (nicht bei RS485 Version)
- Diskette/CD mit Treibern
- Installations-Kurzanleitung

Verfügbare Varianten

- USB4-WM-FTT mit integriertem FTT-10A Transceiver
- USB4-RJ-485 mit integriertem RS485 Transceiver



Weitere Informationen zu LonWorks®-Netzwerken finden Sie unter:

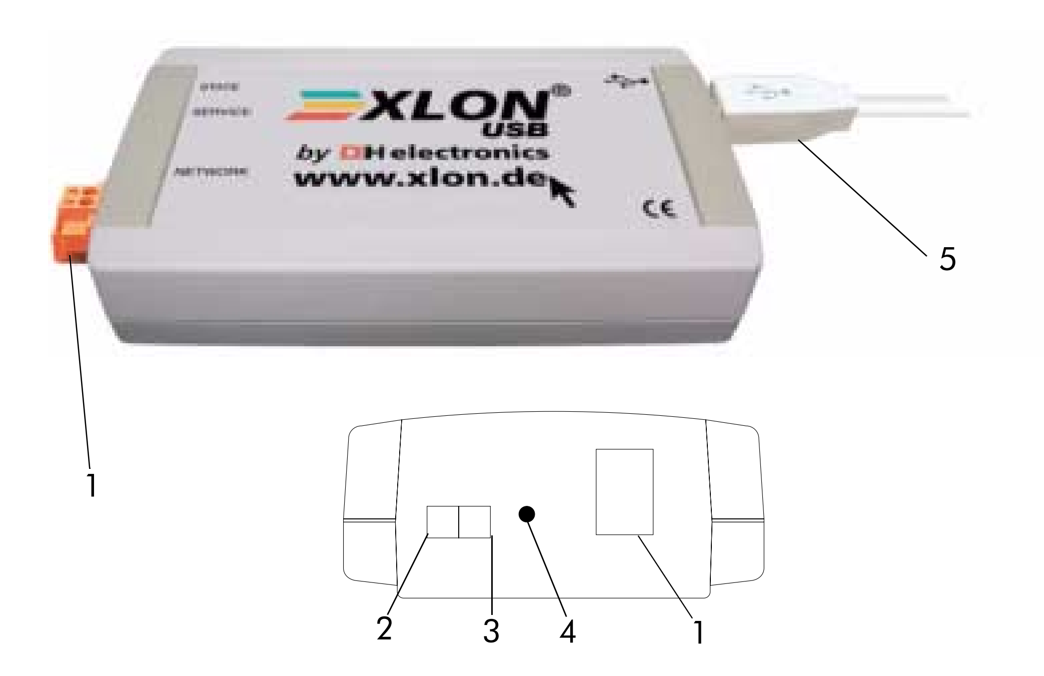
www.echelon.com

3 Installation des XLON® USB Adapters

3.1 Hinweise

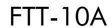
Der PC muss zur Hardwareinstallation des XLON® USB Adapters nicht heruntergefahren werden.

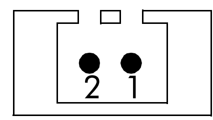
3.2 Hardwareinstallation



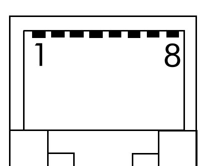
Nr.	Bezeichnung	Bemerkung
1	Anschlußbuchse LON	2-polig: FTT-10A Transceiver Variante RJ45: RS485 Transceiver Variante
2	LED grün „STATE“	Aus: Nicht betriebsbereit Ein: Betriebsbereit
3	LED gelb „SERVICE“	Anzeige Service Pin Neuron Prozessor
4	Service Pin Taster	Manuelles Auslösen der Service Pin Meldung
5	Anschluß USB	Bitte mitgeliefertes Kabel verwenden!

Pinbelegung Anschlußbuchse LON









Pin	FTT-10A	RS485
1	NET B	RS485 A
2	NET A	RS485 B
4	nicht vorhanden	GND

- Den XLON USB Adapter über das mitgelieferte USB-Kabel an einen freien USB Port anschließen
- LonWorks®-Netzwerk-Kabel anstecken
- Bei Windows basierenden Systemen startet der „Assistent für das Suchen neuer Hardware“, siehe Kapitel 3.3.1
- Zur Installation der Gerätetreiber unter Windows CE ist entsprechend Kapitel 3.3.2 vorzugehen

Deinstallation

Stecken Sie den XLON USB Adapter vom USB Port ab. Eine Deinstallation der Treibersoftware ist nicht erforderlich.

3.3 Software- / Treiberinstallation

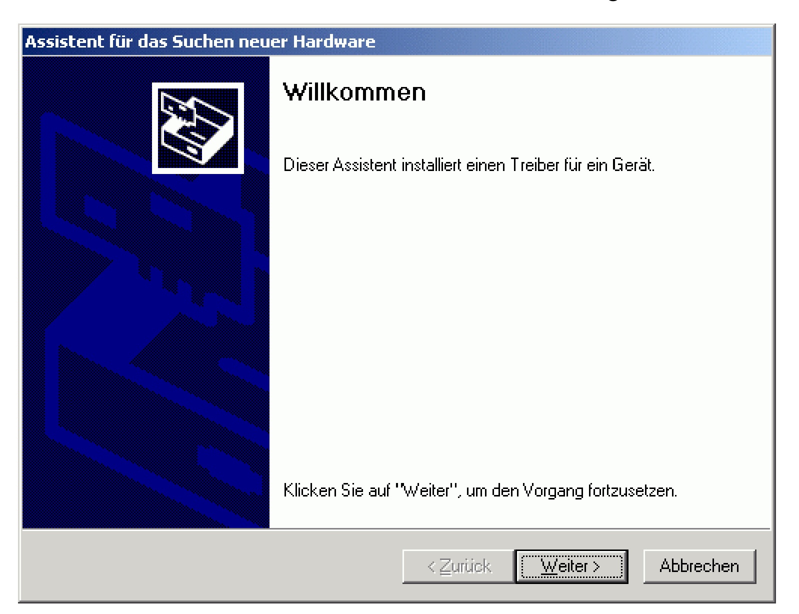
3.3.1 Neuinstallation unter Windows

Als Beispiel wird im Folgenden die Neuinstallation unter Windows 2000 erklärt. Die Installation unter den anderen Windows-Betriebssystemen erfolgt sinngemäß in gleicher Weise!

Nach dem Anstecken des XLON USB Adapters erscheint folgende Meldung:



Der Assistent für das Suchen neuer Hardware wird automatisch gestartet:



- Klicken Sie auf „Weiter>"



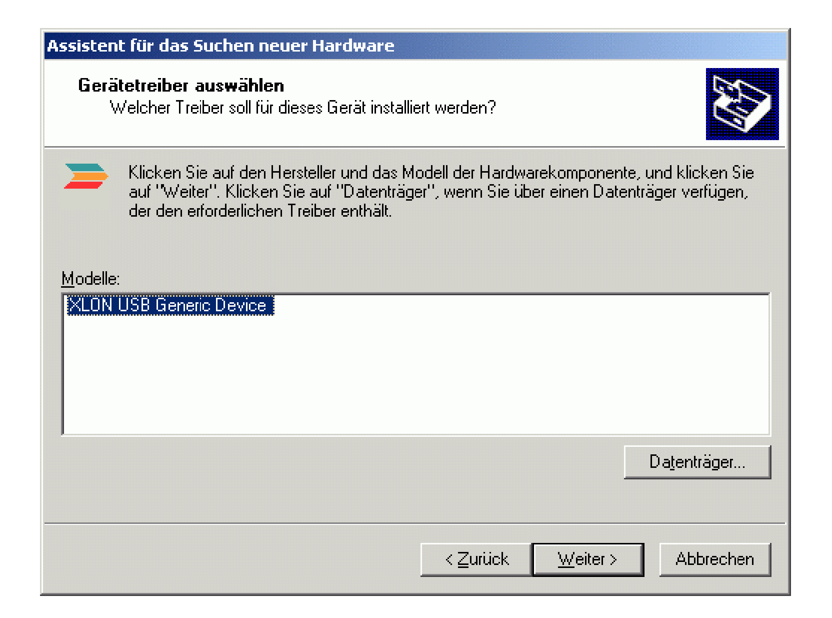


- Übernehmen Sie die oben abgebildete Vorgehensweise und klicken Sie auf „Weiter>“

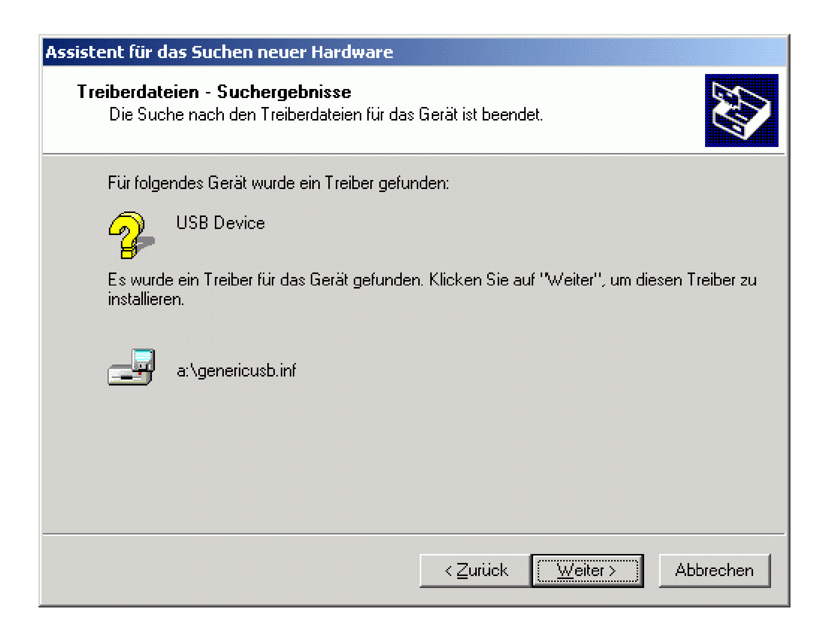


- Wählen Sie, abhängig vom Speichermedium das Ihrem Gerät beigefügt ist, „Diskettenlaufwerke“ bzw. „CD-ROM-Laufwerke“ aus
- Legen Sie die Diskette bzw. die CD-ROM in das entsprechende Laufwerk ein
- Klicken Sie auf „Weiter>“

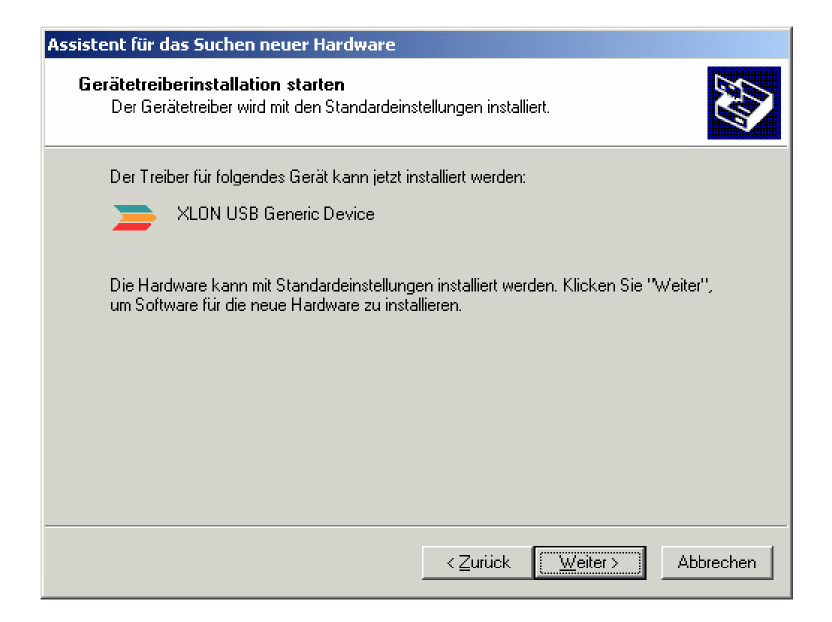
Ein Treiber wurde gefunden:



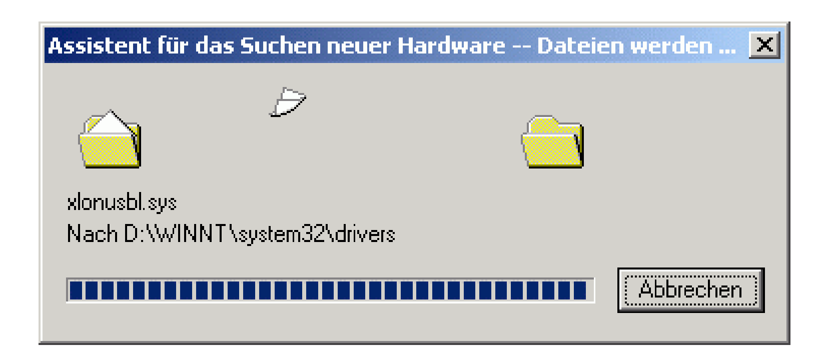
- Klicken Sie auf „Weiter>“



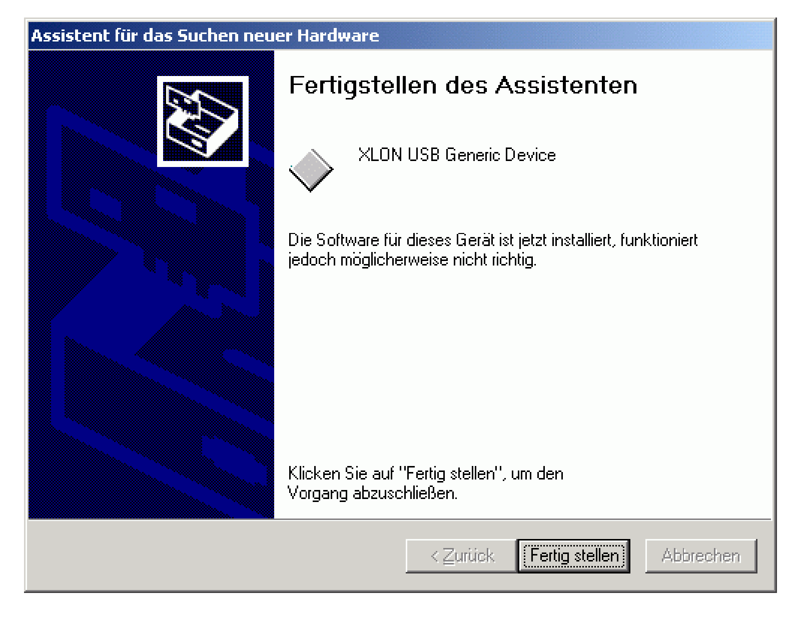
- Klicken Sie auf „Weiter>“



- Klicken Sie auf „Weiter>"



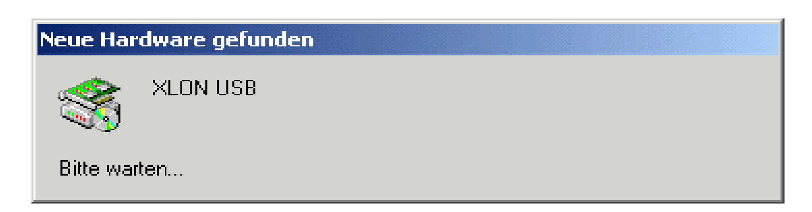
Nach Beendigung des Installationsvorganges:



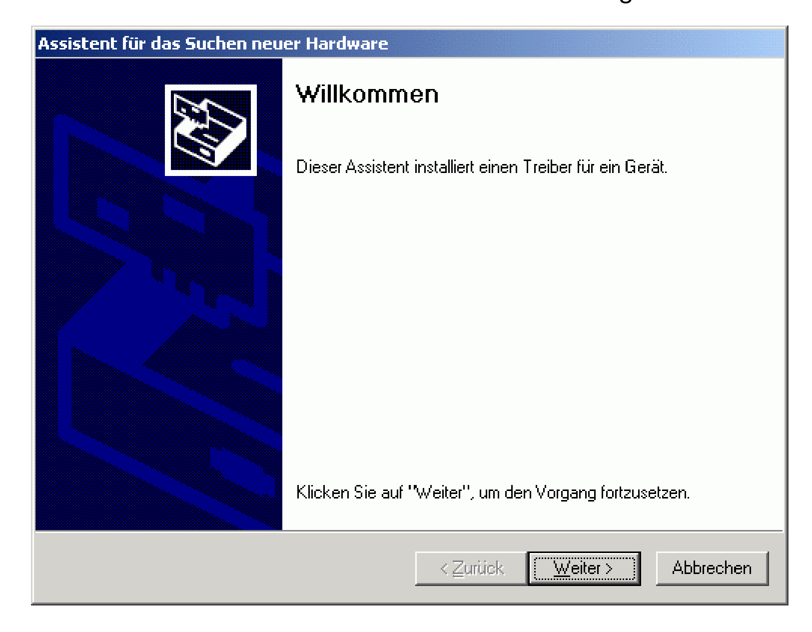
- Klicken Sie auf „Fertig stellen"



Nach dem Installieren des XLON® USB Generic Gerätetreibers muss ein zweiter Gerätetreiber installiert werden. Es erscheint folgende Meldung:



Der Assistent für das Suchen neuer Hardware wird automatisch gestartet:



- Klicken Sie auf „Weiter>“



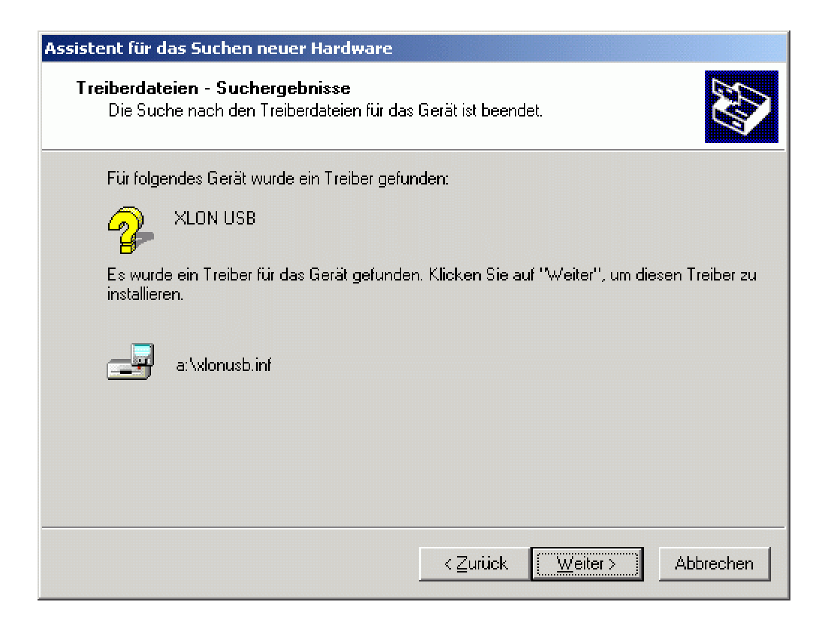
- Übernehmen Sie die oben abgebildete Vorgehensweise und klicken Sie auf „Weiter>“

- Wählen Sie, abhängig vom Speichermedium das Ihrem Gerät beigefügt ist, „Diskettenlaufwerke“ bzw. „CD-ROM-Laufwerke“ aus
- Legen Sie die Diskette bzw. die CD-ROM in das entsprechende Laufwerk ein
- Klicken Sie auf „Weiter>“

Ein Treiber wurde gefunden:



- Klicken Sie auf „Weiter>“



- Klicken Sie auf „Weiter>“





Nach Beendigung des Installationsvorganges:



- Klicken Sie auf „Fertig stellen“

3.3.2 Neuinstallation unter Windows CE 3.0

Der Gerätetreiber für Windows CE 3.0 ist in Form eines „Stream Interface Device Drivers“ als Dynamic Link Library (DLL) für die folgenden Prozessor-Plattformen implementiert:

- ARM
- MIPS
- SH3
- SH4
- x86

Die zur Installation unter Windows CE 3.0 benötigten Dateien können von der Website www.xlon.de heruntergeladen werden.

Der Gerätetreiber unterstützt bis zu 127 XLON USB Adapter in einem System.

Der Gerätetreiber kann entweder dynamisch zu einem laufenden Windows CE System hinzugefügt werden oder statisch in ein neu zu erstellendes Windows CE Image eingebunden werden. Letzteres sollte nur von erfahrenen Anwendern durchgeführt werden, die ein neues Windows CE 3.0 Image erzeugen wollen. Beide Vorgehensweisen sind im folgenden beschrieben.

3.3.2.1 Hinzufügen zu laufendem Windows CE System

Zur Installation des Treibers bei einem Windows CE Device mit statischem RAM sind die Treiberdateien „xlon_usbl.dll“ und „xlon_usb.dll“ manuell in das Verzeichnis „\Windows“ zu kopieren. Anschließend sind die Registrierungseinträge, wie im Kapitel 3.3.2.3 beschrieben, anzulegen.



3.3.2.2 Erstellen eines neuen Windows CE Images

Um den Gerätetreiber für den XLON USB Adapter im „Microsoft Platform Builder 3.0" verfügbar zu machen, sind die folgenden Schritte nötig. Diese beziehen sich auf eine x86 Hardware Plattform, die Vorgehensweise bei anderen Hardwarearchitekturen erfolgt analog, allerdings können sich plattformspezifische Verzeichnispfade unterscheiden.

- Kopieren Sie die Gerätetreiberdateien „xlon_usbl.dll" und „xlon_usb.dll" ins Verzeichnis „_WINCEROOT\PLATFORM\CEPC\FILES".
- Kopieren Sie die Komponentendatei „xlon.cec" ins Verzeichnis „CEPBDir\CEPB\ CEC".
- Starten Sie „Microsoft Platform Builder 3.0" und laden Sie ihren Plattform Arbeitsbereich.
- Öffnen Sie das Menü „File" und wählen Sie „Manage Platform Builder Components".
- Importieren Sie die Komponentendatei „xlon.cec" indem Sie auf „Import New" klicken.
- Fügen Sie im „Platform Builder" die Registrierungsinformationen wie unter Kapitel 3.3.2.3 beschrieben manuell zur Datei „platform.reg" hinzu.
- Fügen Sie den Inhalt der Datei „xlon_usb.bib" manuell im „Platform Builder" zur Datei „platform.bib" hinzu.
- Öffnen Sie im „Platform Builder" das Menü „View", und klicken Sie auf „Catalog", anschließend sollte sich die Katalog-Ansicht öffnen.
- Öffnen Sie in der Baumansicht des Katalogs den Zweig „Catalog/Drivers/CEPC/XLON".
- Fügen Sie über die rechte Maustaste und „Add to Platform" die „XLON USB" Komponenten (xlonusbl und xlonusb) zur aktuellen Plattform hinzu.
- Starten Sie den „Platform Builder" neu, erzeugen Sie Ihr neues Windows CE 3.0 Image und laden Sie es auf Ihr Zielsystem. Die Gerätetreiber für den XLON USB Adapter werden durch das „USB Subsystem" von Windows CE nach dem Booten des Zielsystems automatisch geladen und stehen für Ihre Applikationen zur Verfügung.

3.3.2.3 Registrierungseintrag

Ein Beispiel für einen korrekten Registrierungseintrag ist in der Datei „xlon_usb.reg" zu finden. Eine Erläuterung des Inhalts wird im folgenden Abschnitt gegeben.

[HKEY_LOCAL_MACHINE\Drivers\USB\LoadClients]
Der Registrierungsschlüssel HKEY_LOCAL_MACHINE\Drivers\USB\LoadClients enthält Unterschlüssel mit denen festgelegt wird, welchen Gerätetreiber das USB Subsystem für welches USB Gerät lädt. Wird nach dem Anstecken eines USB Geräts ein zugehöriger Registrierungseintrag gefunden, so wird der zugehörige Treiber geladen, ansonsten wird der Benutzer zur Eingabe eines Treibers aufgefordert.

Die Unterschlüssel von „LoadClients" haben die Form „Group1_ID\Group2_ID\Group3_ID\ Driver Name", wobei sich die Zeichenkette mit dem Namen der Treiberdatei unter dem letzten Schlüssel „Driver Name" befindet. Für den XLON USB Adapter ist nur die „Group1_ID" von Bedeutung, diese spezifiziert die Vendor- und Product-ID des Geräts, getrennt durch einen Underscore. Die „Group2_ID" spezifiziert die Deviceklasse, die „Group3_ID" die Interfaceklasse die vom Treiber unterstützt wird. Da der XLON USB Adapter durch die „Group1_ID" eindeutig gekennzeichnet ist, sind die „Group2_ID" und „Group3_ID" auf den Wert „Default" gesetzt, d.h. der XLON USB Treiber wird immer dann geladen, wenn ein USB Gerät mit der passenden Vendor- und Product-ID am USB angesteckt wird. Der Unterschlüssel „Driver Name" trägt den Namen des jeweiligen Gerätetreibers und enthält die in der folgenden Tabelle beschriebenen Einträge.

.

Eintrag	Typ	Beschreibung
Dll	**REG_SZ**	Dieser Eintrag spezifiziert den Dateinamen der Gerätetreiber-DLL, die der Device Manager lädt, hier z.B. xlon_usbl.dll oder xlon_usb.dll.

3.3.2.4 Hardwarekonfiguration

Mit dem Windows CE 3.0 Gerätetreiber können bis zu 127 XLON® USB Adapter in einem System verwendet werden. Eine weitergehende Hardwarekonfiguration ist durch die Hot-Plug and Play Fähigkeit von USB Systemen nicht notwendig.

3.3.3 Neuinstallation unter Linux

Derzeit ist für den XLON® USB Adapter kein Linux Treiber verfügbar.

3.3.4 Treiberupdate

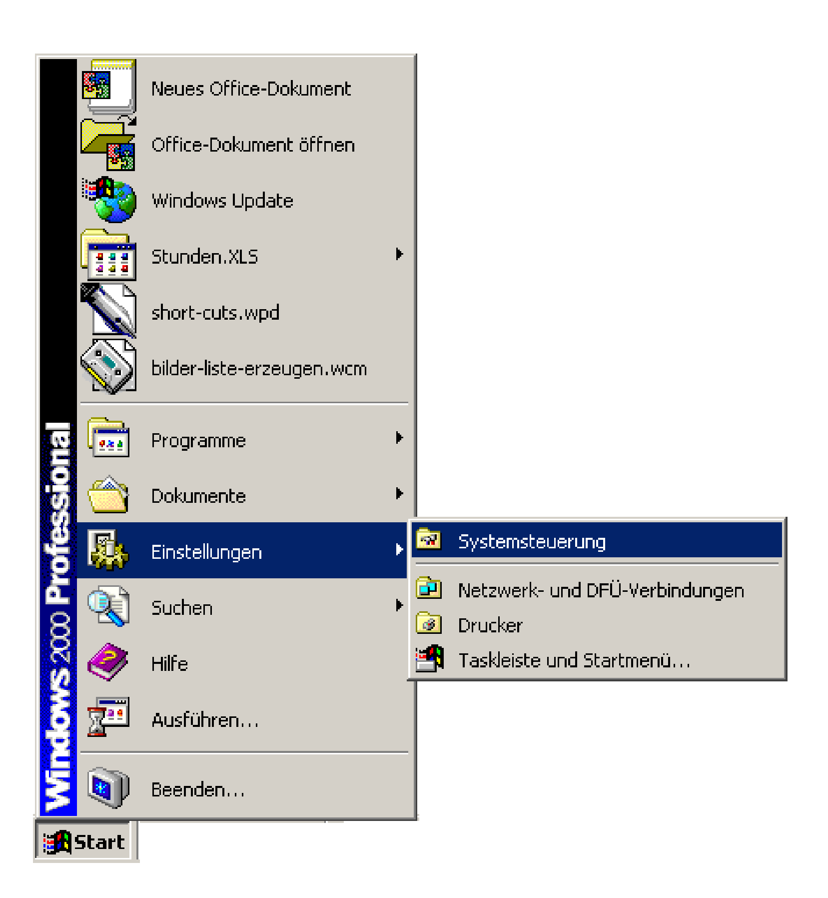
☞ Die neuesten Treiber zum Download finden Sie unter:

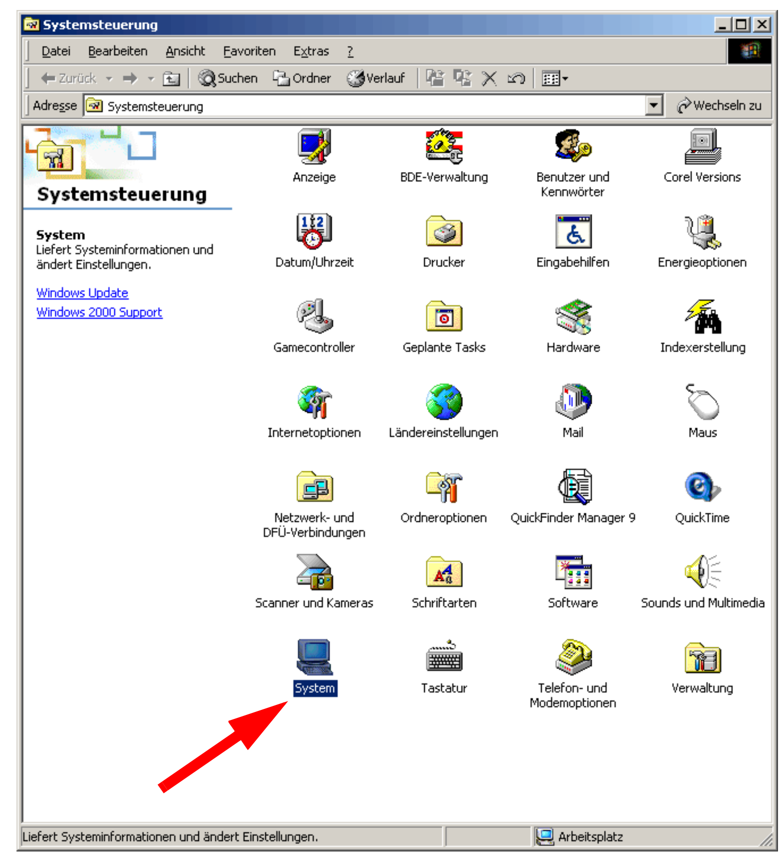
www.xlon.de

- Laden Sie den für Ihr Betriebssystem passenden Treiber herunter und speichern ihn an beliebiger Stelle

3.3.4.1 Windows

- Öffnen Sie die Systemsteuerung
  - Klicken Sie auf „START“
  - Gehen Sie mit dem Mauszeiger auf „Einstellungen“
  - Klicken Sie auf „Systemsteuerung“



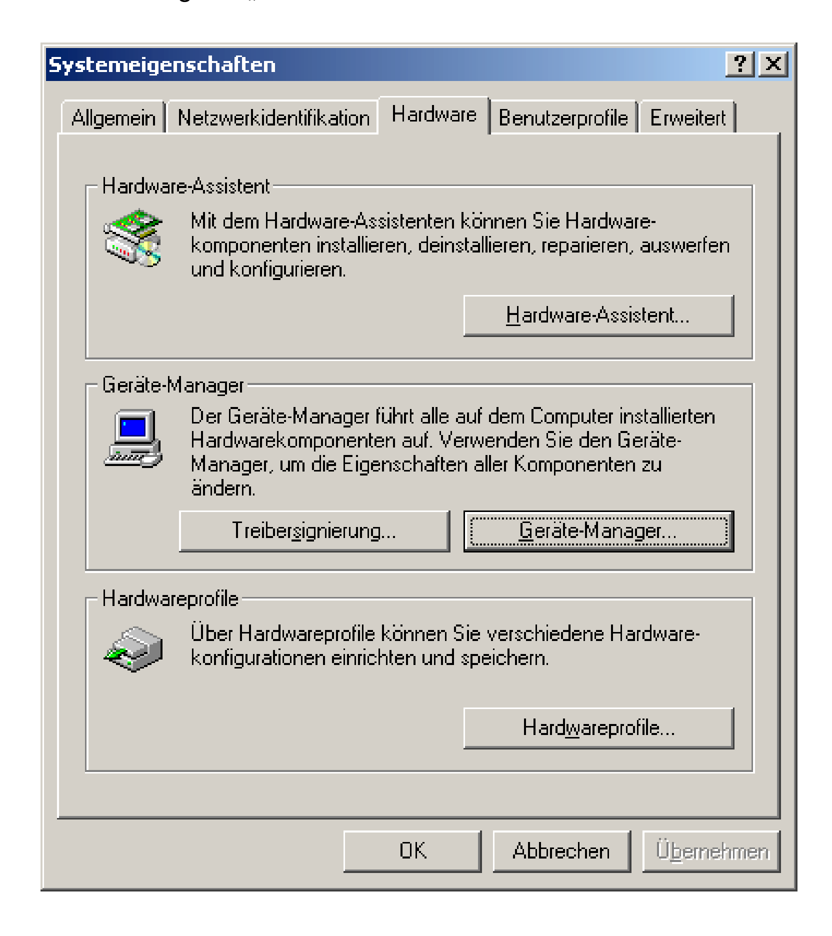


- Wählen Sie „System“ durch Doppelklick aus

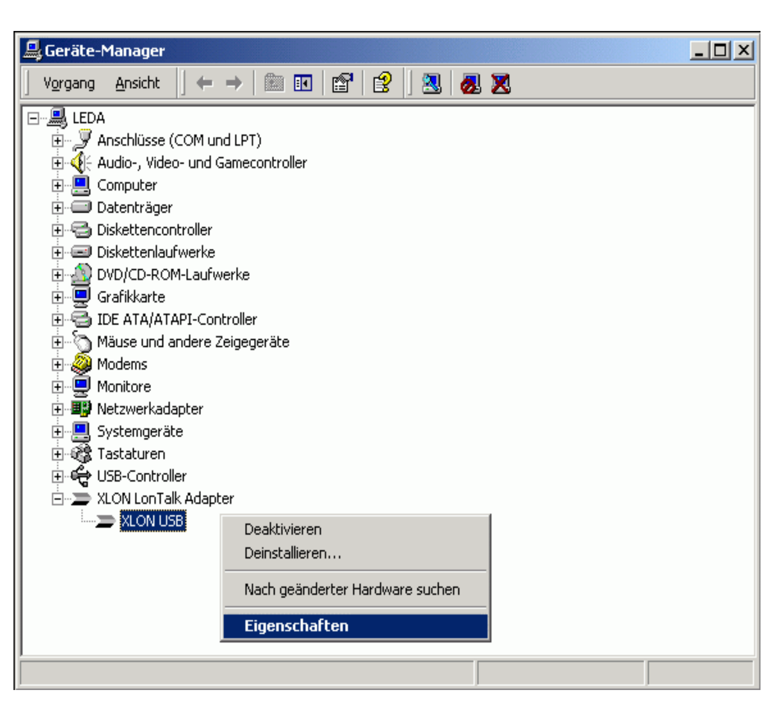


de

- Gehen Sie auf das Register „Hardware“

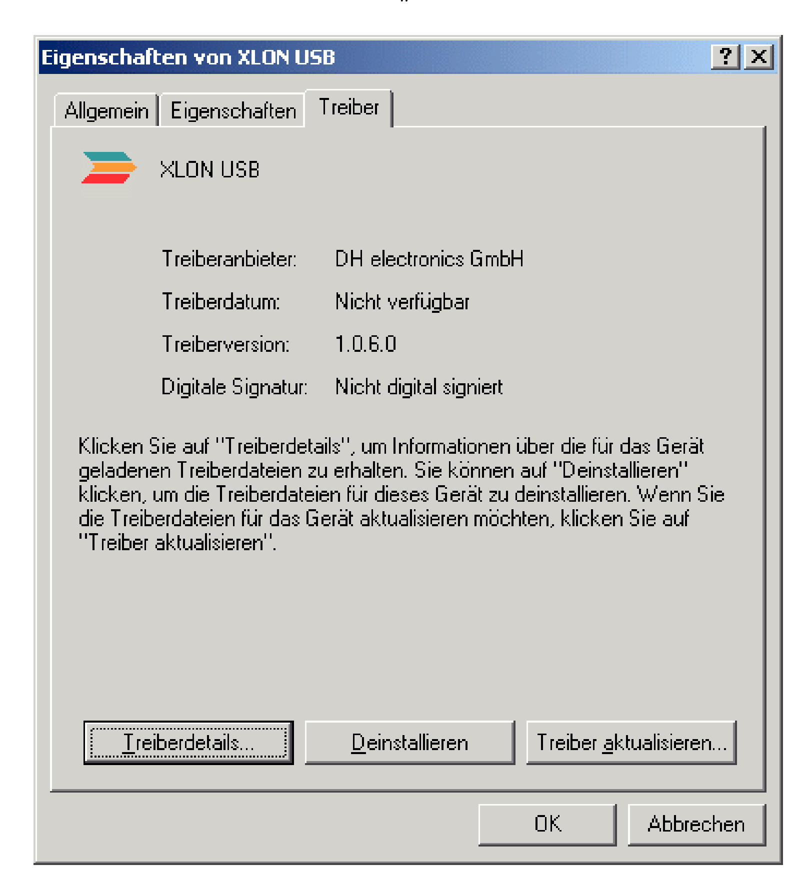


- Öffnen Sie den „Geräte-Manager...“ durch Anklicken mit der Maus



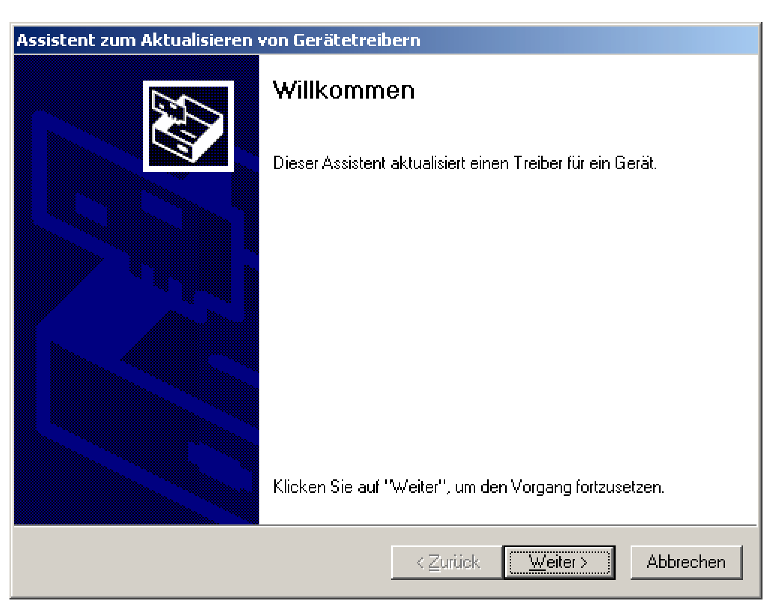
- Öffnen Sie das „Eigenschaften“-Fenster des XLON USB Adapters (rechte Maustaste)

- Klicken Sie auf den Reiter der Karteikarte „Treiber“



- Prüfen Sie, ob die auf Ihrem System installierte Treiberversion niedriger ist, als die Version, die Sie installieren möchten.
- Klicken Sie auf „Treiber aktualisieren...“

- Der Assistent zum Aktualisieren von Gerätetreibern öffnet sich
- Folgen Sie den Anweisungen am Bildschirm



- Klicken Sie auf „Weiter>“

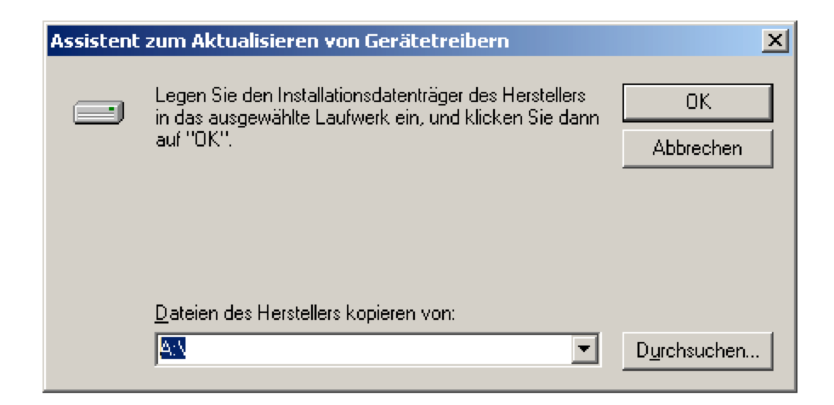


- Klicken Sie auf „Weiter>“

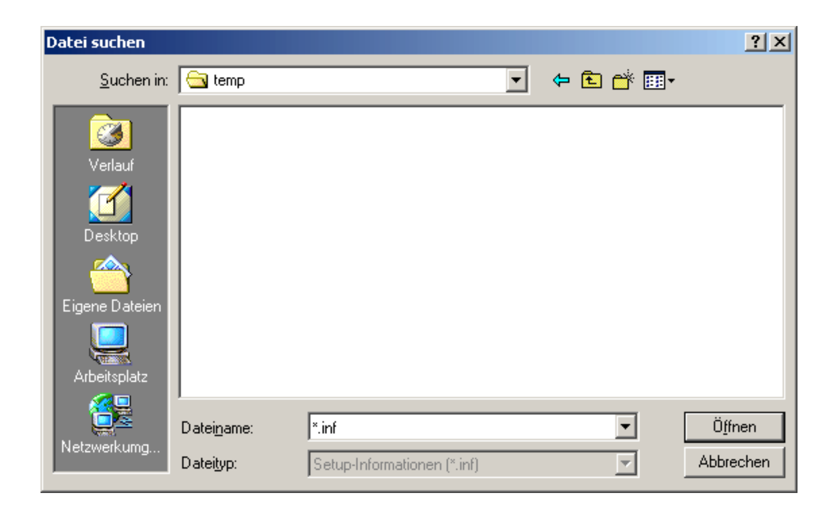


- Wählen Sie „Andere Quelle angeben“ aus und alle anderen Möglichkeiten ab
- Klicken Sie auf „Weiter>“





- Klicken Sie auf „Durchsuchen...“



- Suchen Sie im Explorerfenster den heruntergeladenen Treiber
- Markieren Sie den Treiber und klicken Sie auf „Öffnen“
- Die weitere Treiberaktualisierung verläuft identisch zur Neuinstallation

3.3.4.2 Windows CE 3.0

Das Treiberupdate entspricht einer Neuinstallation wie in Kapitel 3.3.2 beschrieben.

3.3.4.3 Linux

Derzeit ist für den XLON® USB Adapter kein Linux Treiber verfügbar.

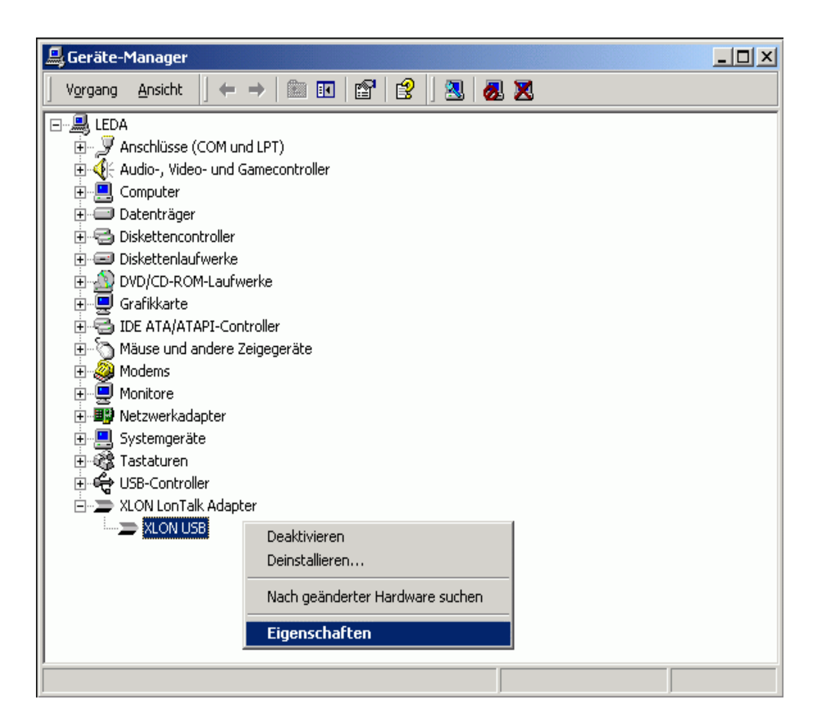
3.4 Deinstallation

Eine Deinstallation der Software ist nicht erforderlich. Es genügt das Abstecken des XLON® USB Adapters. Informationen hierzu finden Sie in Kapitel 3.2.



4 Inbetriebnahme und Test

- Starten Sie den Geräte-Manager und öffnen Sie das „Eigenschaften"-Fenster des XLON® USB Adapters, siehe Kapitel 3.3.4



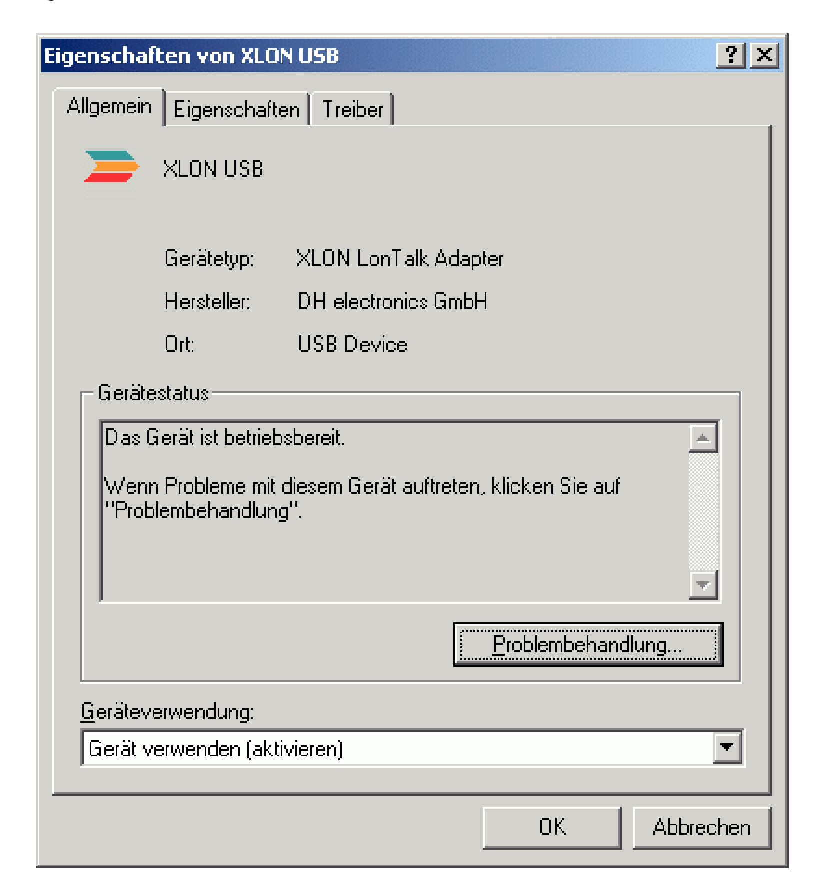
- Das „Eigenschaften"-Fenster öffnet sich



4.1 Überprüfen der Einstellungen unter Windows

4.1.1 Allgemeine Einstellungen

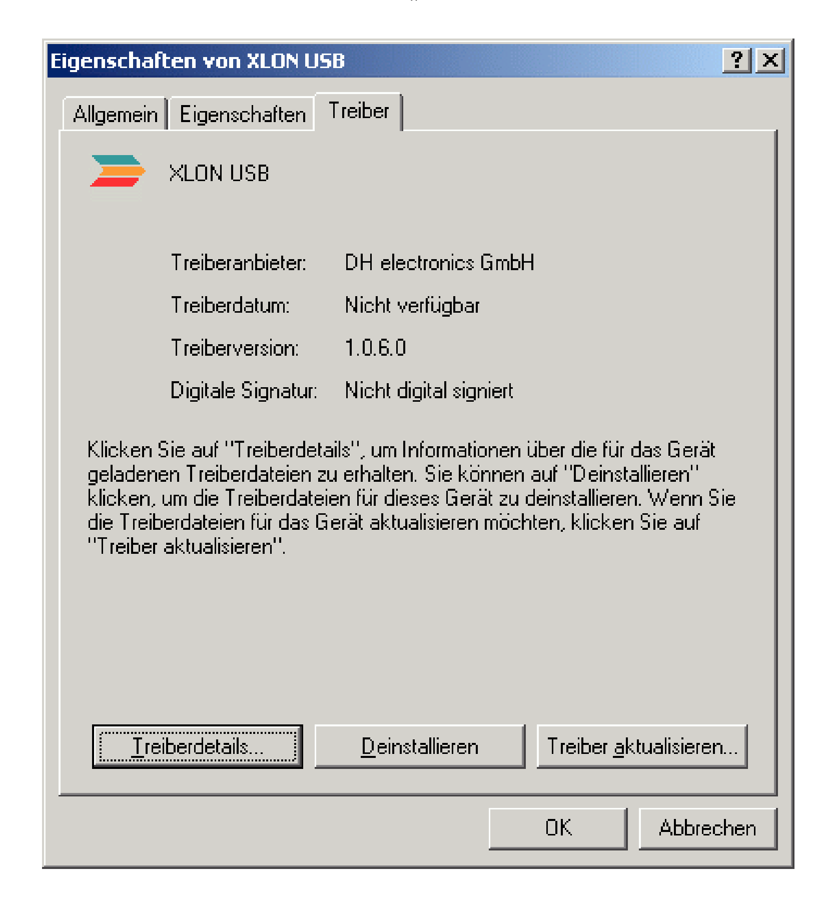
Karteikarte Allgemein:



☞ Wichtig: Im Feld „Gerätestatus" muß „Das Gerät ist betriebsbereit." zu lesen sein!

4.1.2 Treiberinformationen

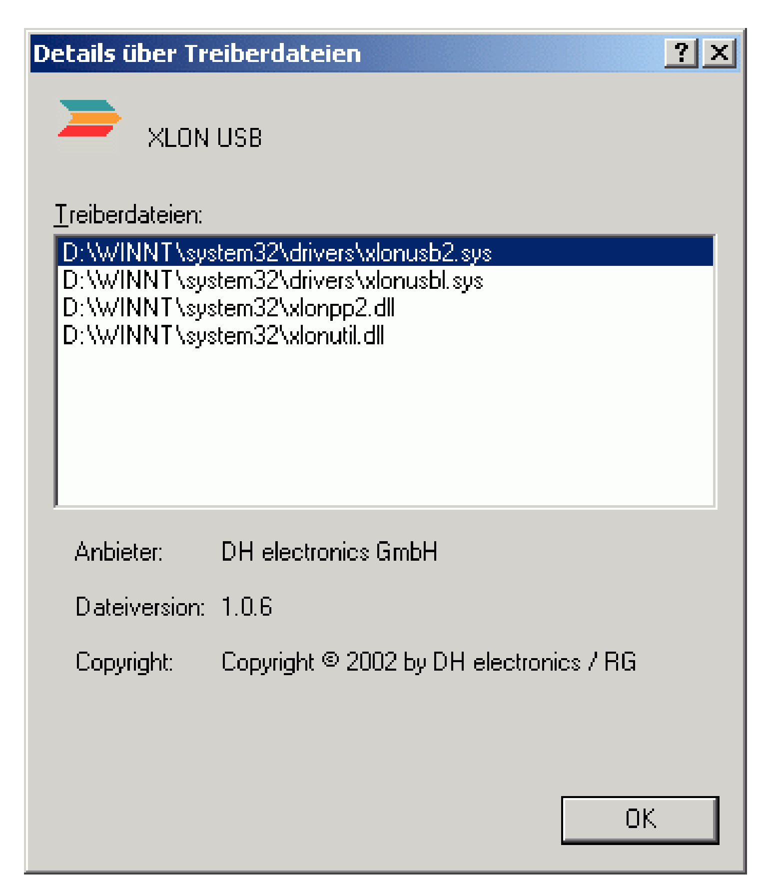
- Klicken Sie auf den Reiter der Karteikarte „Treiber"



Die Daten des aktuellen Treibers werden angezeigt.

☞ Die angezeigte Versionsnummer hinter dem Begriff „Treiberversion" ist nicht die Versionsnummer der Treiberdatei, sondern lediglich die Nummer der Treiberinstallation!

- Um die Versionsnummer des Treiber zu erfahren, klicken Sie auf „Treiberdetails...“



Bei Auswahl einer der angezeigten Dateien erhalten Sie Informationen zum jeweiligen Anbieter, der Dateiversion und dem Copyright.

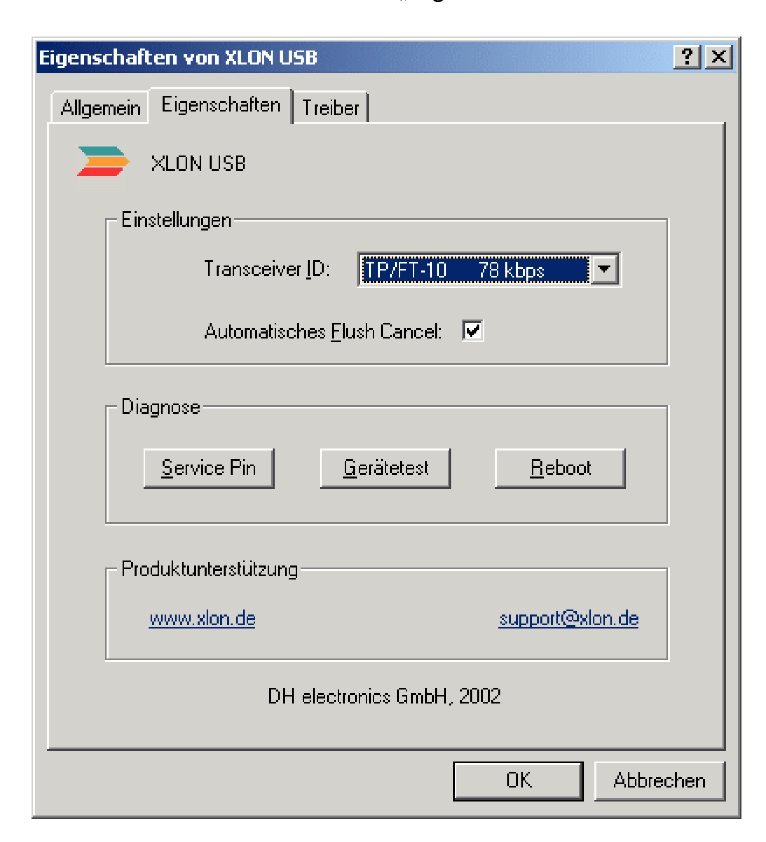
☞ **Diese Informationen sollten Sie für Supportanfragen immer bereit halten!**

Informationen zu den Buttons „Deinstallieren“ und „Treiber aktualisieren...“ finden Sie im Kapitel 3.3



4.1.3 Eigenschaften des XLON USB Adapters

- Klicken Sie auf den Reiter der Karteikarte „Eigenschaften"



☞ Die aktuelle Konfiguration des XLON USB Adapters für „Transceiver ID" und „Automatic Flush Cancel" wird angezeigt.

Im Auslieferungszustand sind folgende Einstellungen vorgenommen:

Transceiver ID: **TP/FT-10 78kbps**

Automatic Flush Cancel: **aktiviert**

Ändern der Transceiver ID

- Stellen Sie sicher, dass mögliche Transceiver-Einstellungen mit der Hardware übereinstimmen. Der XLON USB Adapter wird bei einer falschen Einstellung nicht funktionieren!
- Klicken Sie auf die „Dreiecks-Taste" (▾) neben dem Anzeigenfeld
- Wählen Sie in dem sich öffnenden Auswahlfenster die gewünschte ID mit der linken Maustaste aus
- Bestätigen Sie durch Anklicken des „OK"-Buttons



Automatic Flush Cancel Ein-/Ausschalten

Nach jedem Reset des XLON USB Adapters ist standardmäßig die Kommunikation über das LonWorks®-Netzwerk gesperrt. Der Empfang bzw. das Senden von Daten ist erst möglich, nachdem ein Flush Cancel Kommando an den XLON USB Adapter gesendet wurde.

Ist die Automatic Flush Cancel Funktion eingeschaltet, so schickt der Gerätetreiber automatisch nach jedem Reset ein Flush Cancel Kommando an den XLON USB Adapter. Eine Kommunikation über das LonWorks®-Netzwerk ist dann möglich.

☞ **Bitte beachten Sie, dass bei ausgeschalteter Automatic Flush Cancel Funktion ein Flush Cancel Kommando von der Anwendung an den XLON USB Adapter gesendet werden muß!**

- Klicken Sie mit der linken Maustaste in das Feld neben „Automatic Flush Cancel"
- ☑: eingeschaltet
- ☐: ausgeschaltet
- Bestätigen sie durch Anklicken des „OK"-Buttons

Die Funktionalität der Buttons „Service Pin", „Gerätetest" und „Reboot" werden in Kapitel 4.2 erklärt.

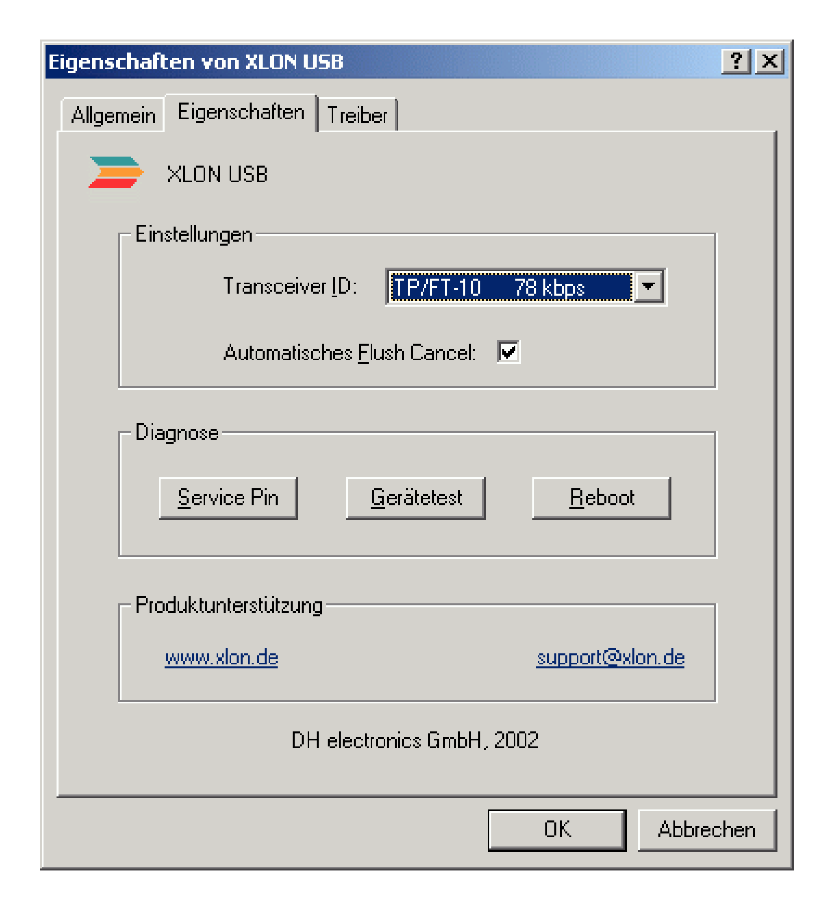
Benötigen Sie zusätzliche Informationen oder Produktunterstützung, verwenden Sie bitte die unter „Produktunterstützung" angegebenen Links.



4.2 Testen des XLON® USB Adapters unter Windows

4.2.1 Diagnose mittels Software

- Öffnen Sie die „Eigenschaften"-Seite des XLON® USB Adapters wie in Kapitel 4 beschrieben
- Öffnen Sie dann die Karteikarte „Eigenschaften"



Für den Gerätetest finden Sie unter der Gruppe „Diagnose" die drei Buttons:

Service Pin:

Durch Betätigen des Buttons „Service Pin" sendet der XLON® USB Adapter eine Service Pin Meldung auf das LonWorks®-Netzwerk. Dies hat exakt die selbe Funktionalität wie die manuelle Betätigung der Service Pin Taste (siehe Kapitel 3.2).

Gerätetest:

Mittels des Buttons „Gerätetest" kann ein Gerätetest durchgeführt werden:

Nach Betätigen des Buttons muß die gelbe Service-LED des Gerätes kurz blinken. War die Kommunikation mit dem XLON® USB Adapter erfolgreich, so wird dies angezeigt.

Reboot:

Das Betätigen des „Reboot" Buttons führt zum Zurücksetzen des XLON® USB Adapters in den Auslieferungszustand. Diese Aktion sollte nur von erfahrenen Benutzern durchgeführt werden.



de

4.2.2 Diagnose mittels Leuchtdioden

Wie in Kapitel 3.2 beschrieben, besitzt der XLON USB Adapter zwei Visualisierungsleuchtdioden. Eine grüne Status-Leuchtdiode und eine gelbe Service-Pin-Leuchtdiode.

Grüne Status-Leuchtdiode:

Die grüne LED zeigt an, ob das Gerät betriebsbereit ist oder nicht. Leuchtet die LED ist das Gerät betriebsbereit und die Treiber sind richtig geladen.

☞ Ist diese Leuchtdiode aus, ist das Gerät nicht betriebsbereit, oder es ist noch kein Gerätetreiber installiert bzw. geladen. Der XLON USB Adapter kann sich auch im Suspend (=Stromspar) Modus befinden.

Gelbe Service-Pin-Leuchtdiode:

Die gelbe LED visualisiert den Zustand der Neuron Prozessor Service Pin Leitung.

- Ist der Gerätetreiber richtig installiert für den XLON USB Adapter richtig installiert, so ist die Leuchtdiode aus.
- Blinkt die LED mit ½ Hz so ist der Zustand des XLON USB Adapters „Unconfigured“.
- Bei Durchführung eines Gerätetests aus der „Eigenschaften“-Seite (siehe Kapitel 4.2.1), oder beim Ausführen eines „Reset“-Kommandos von einer Anwendung aus blinkt die Leuchtdiode kurz auf.

☞ Blitzt die LED mit 1,25 Hz und kann eine Anwendung nicht auf den XLON USB Adapter zugreifen, so liegt ein Problem in der Hardware vor.

Schlägt der Zugriff aus einer Anwendung fehl und verändert sich der Zustand der Leuchtdiode nicht, so ist vermutlich kein Gerätetreiber installiert.

Gelbe Service Pin LED	Beschreibung
Konstant aus	- XLON USB Adapter ist richtig installiert und betriebsbereit wenn, wenn grüne LED konstant an ist oder - XLON USB Adapter ist nicht richtig installiert wenn grüne LED konstant aus ist.
Konstant ein	- Ein Hardwaredefekt liegt vor
Blinkt mit ½ Hz	- Zustand des XLON USB Adapters ist „Unconfigured“, d.h. der XLON USB Adapter hat keine Netzwerkadresse
Blitzt 1 mal kurz auf	- ein „Reset“ auf den XLON USB Adapter wurde ausgeführt oder - ein Gerätetest wurde durchgeführt



5 Technische Informationen

5.1 Hardware

5.1.1 Allgemeine Informationen

Bus Anschluß	USB konform, gemäß USB Spezifikation Revision 1.1, 12 Mbit/s
Netzwerkanschluß FTT-10A: RS485:	2-poliger Stecker (Weidmüller) mit Schraubklemmen und Zugentlastung Western Modular (RJ45) Steckverbinder 8 polig
Stromversorgung	Erfolgt über USB
Service-Pin-Funktion	Gesteuert vom Host-Rechner oder durch externe Service-Pin-Taste
Konfigurations-Status	Anzeige auf Host-Rechner und über Service-LED
Netzwerk-Transceiver	wahlweise FTT-10A oder galvanisch isolierter RS485 (integriert)
Netzwerk-Topologien	FTT-10A: Free Topology und Link Power RS485: Twisted Pair
Daten für Stromversorgung	5 V DC, ±5%, 100 mA typisch
Betriebstemperatur	0°C bis +70°C
Lagertemperatur	-45°C bis +85°C
maximale Luftfeuchtigkeit	90% bei +50°C, nicht kondensierend
EMV-Richtlinien	EN55022 Level B, EN61000-4-2, EN61000-4-4, EN50140, EN50141
Prüfzeichen	CE und FCC
Prozessor	Neuron® 3150 - Prozessor mit 10 MHz
Abmessungen	123 x 68 x 30 mm (4.84“ x 2.68“ x 1.18“) (Länge x Breite x Höhe)
Gewicht	100 g

Die Hardware des XLON® USB Adapters unterstützt bis zu 127 Geräte pro USB System im PC (Multiple Device Unterstützung).



5.1.2 Steckverbinder

Als Steckverbinder für den LonWorks®-Anschluß des XLON USB Adapters kommt bei der FTT-10A Varante ein Steckverbinder der Serie BL3.5 der Firma Weidmüller zum Einsatz. Bei der RS485 Variante kommt ein Western Modular (RJ45) Steckverbinder zum Einsatz

Netzwerktyp	Polzahl	Weidmüller Bezeichnung	Bestellnummer
Free Topologie FTT	2 polig	BL3.5/2F SN OR	160664

Der maximal einklemmbare Leitungsquerschnitt beträgt beim Weidmüller Steckverbinder 1,5 mm^2.

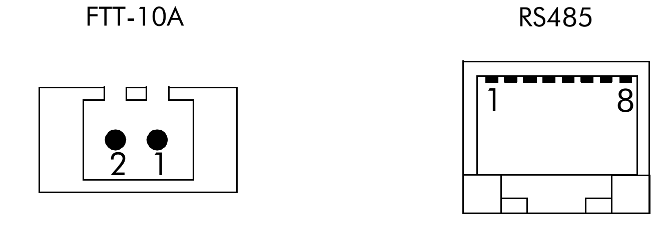
☞ Bestellinformationen sowie weitere, detaillierte Informationen für diese Buchsenleiste finden Sie unter:

www.weidmueller.de

Steckerbelegung:

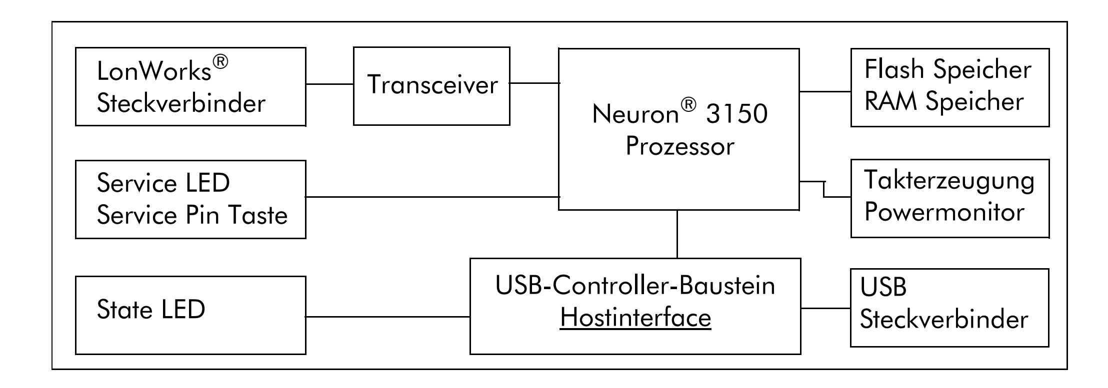
Pinbelegung Anschlußbuchse LON

FTT-10A RS485



Pin	FTT-10A	RS485
1	NET B	RS485 A
2	NET A	RS485 B
4	nicht vorhanden	GND

5.1.3 Blockschaltbild



5.1.4 Technische Details der Hardware

Anbindung an das Host System über den USB Bus:

Die Ankopplung an den USB erfolgt gemäß USB-Spezifikation Revision 1.1. Die Übertragungsrate beträgt 12 Mbit/s.

Der XLON® USB Adapter ist vollständig Plug&Play-kompatibel. Er identifiziert sich im System durch die von der ‚USB Implementers Forum' zugewiesenen DH electronics GmbH Vendor ID ‚0x0916' sowie den beiden Device ID's ‚0x0001' und ‚0x0002'.

LonWorks®-Netzwerk Interface:

Für das LonWorks®-Netzwerk Interface stehen zwei verschiedene Transceiver-Varianten zur Verfügung: Free Topology Transceiver FTT-10A (Klemme zweipolig) und RS485 Transceiver (Western Modular Steckverbinder). Die Übertragungsrate beim FTT-10A Transceiver beträgt 78,5 kBit/s. Ist der XLON® USB Adapter mit RS485 Transceiver ausgestattet, so können mittels Software verschiedene Übertragungsraten eingestellt werden (vgl. Kapitel 4.1.3). Die maximale Übertragungsrate beträgt hier 250kBit/s. Die Isolationsspannung beträgt 1,5 kV (UL rated).

Neuron Prozessor Core:

Als Neuron Prozessor kommt ein 3150® Prozessor mit externem Speicherinterface zum Einsatz. Für den Programmspeicher wird wiederbeschreibbarer Flashspeicher und als Datenspeicher wird SRAM Speicher verwendet. Der Neuron Prozessor ist mittels des Neuron ‚Parallel IO Modells' an den USB-Controller-Baustein angekoppelt.

Adresstabelle Neuron Prozessor Core:

Speichertyp	Adressbereiche	Speichergröße
ROM-Speicher	0x0000 - 0xC2FF	49919 Byte (48,75 kB)
RAM-Speicher, lesen und schreiben	0xC300 - 0xE6FF	9215 Byte (9,00 kB)
EA-Bereich für Interrupt-Generierung	0xE700 - 0xE7FF	
Reserviert - Neuron® Prozessor intern	0xE800 - 0xFFFF	

5.1.5 Unterstützte Transceiver

In der Regel braucht die Transceiver-Konfiguration nicht verändert werden. Sollte dies doch nötig werden, geschieht dies unter Microsoft Windows Desktop Betriebssystemen im Gerätemanager. Für andere Betriebssysteme wie Microsoft Windows CE oder Linux kann es nötig werden, die Transceiver ID manuell in bestimmte Konfigurationsdateien einzutragen. Genauere Informationen hierüber sind im Kapitel „Treiberinstallation" für die jeweiligen Betriebssysteme nachzulesen. Die folgende Tabelle zeigt die von der Hardware des XLON USB Adapters grundsätzlich unterstützten LON Transceiver ID's. Abhängig von dem physikalisch auf dem Gerät vorhandenen Transceiver (TP/FT-10 oder TP-RS485) werden nicht alle in der Tabelle angegebenen Transceiver-Betriebsarten unterstützt.

ID	Name	Medium	Netzwerk Übertragungsrate
04	TP/FT-10	Flexible topology/link power	78 kbps
05	TP-RS485-39	RS-485 twisted pair	39 kbps
12	TP-RS485-78	RS-485 twisted pair	78 kbps



6 Softwarezugriff

6.1 Applikationsschnittstelle unter Windows

6.1.1 LNS-Anwendungen

Will man über eine LNS-Anwendung auf den XLON® USB Adapter zugreifen, so muß man lediglich als Netzwerkinterface den XLON® USB Adapter angeben, den man verwenden möchte. Die Gerätetreiber unterstützen bis zu 127 XLON® USB Adapter pro PC-System, d.h. Sie können mehrere XLON® USB Adapter parallel in Ihrem System verwenden. Die Geräte sind über unterschiedliche Gerätenamen ansprechbar. Diese unterscheiden sich in der Netzwerk-Interface-Nummer.

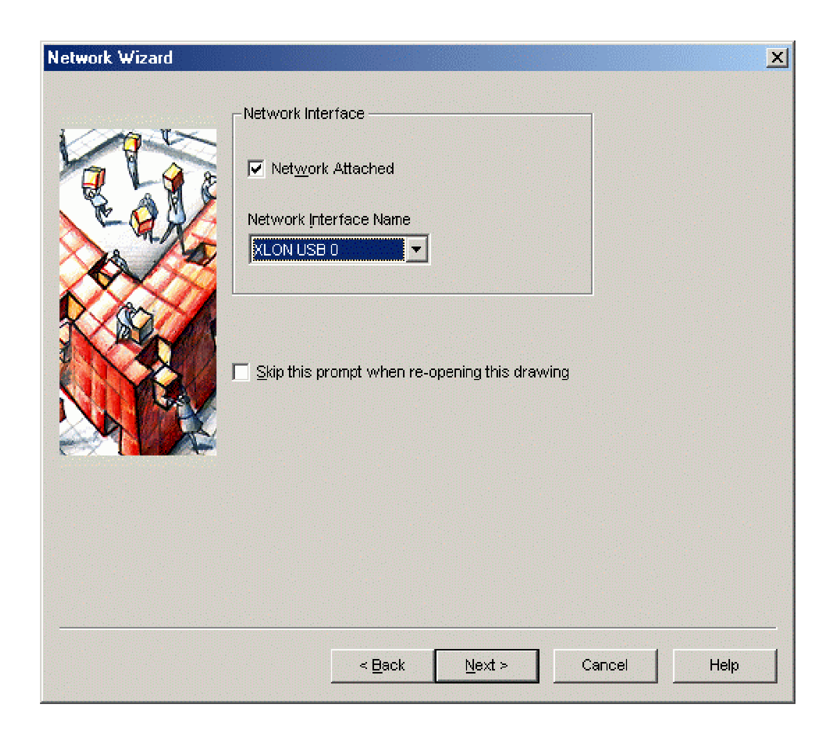
Die LNS-Anwendung gibt ein Auswahlmenü für das Netzwerk-Interface vor. Sie finden den XLON® USB Adapter unter dem Netzwerk-Interface-Namen:

XLON USB x

x: XLON® USB Netzwerk-Interface-Nummer

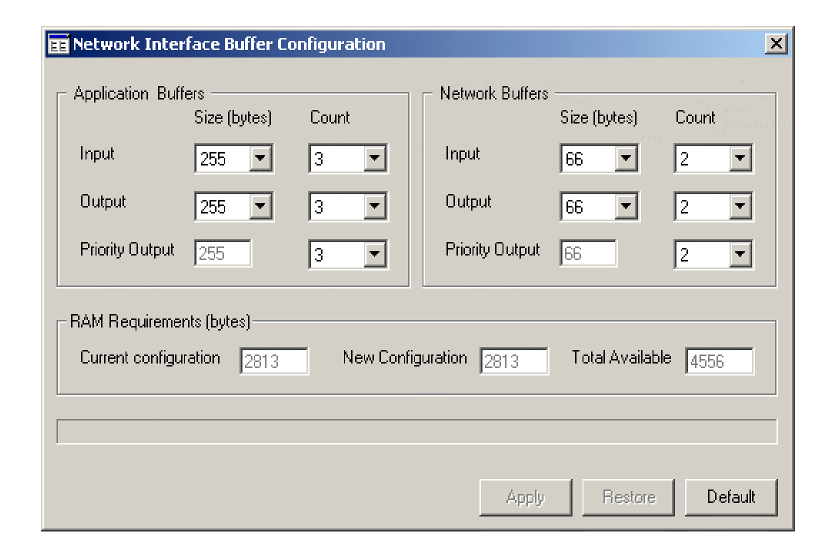
z.B. XLON USB 0 => erster im System eingebaute XLON® USB Adapter.

Beispielhaft zeigt der folgende Bildschirmauszug die Konfiguration in der Anwendung ‚LonMaker for Windows':



6.1.2 Konfiguration der Netzwerk Interface Puffer

Über das LNS Plug-In ‚Network Interface Buffer Configuration' von Echelon (zu finden unter www.echelon.com) kann man die Netzwerk- und Applikations-Puffer des Neuron Prozessors verändern.



6.1.3 Programmierung einer eigenen Anwendung

Basierend auf den Informationen des ‚LonWorks Host Application Programmer's Guide' ist es möglich eine eigene LonWorks Host Anwendung zu programmieren. Diese Programmieranleitung können Sie von der Firma Echelon (www.echelon.com) beziehen.

Wie der Zugriff auf den Gerätetreiber des XLON USB Adapters in C zu kodieren ist, wird in diesem Kapitel erläutert. Da das Echelon Standardtreiberinterface zwischen Windows 98/ME und Windows 2000/XP basierenden Systemen unterschiedlich ist, muß zwischen den beiden Betriebssystemfamilien unterschieden werden.

Alle unten angeführten Funktionen sind Windows 32-Bit API-Funktionen.

☞ Ein ausführliches Programmierbeispiel steht auf der Webseite www.xlon.de zum Download zur Verfügung.

6.1.3.1 Öffnen des Gerätetreibers

Bevor auf den Treiber zugegriffen werden kann muß er geöffnet werden. Das Betriebssystem liefert beim erfolgreichen Zugriff ein Handle zurück, über den anschließend auf den Treiber zugegriffen werden kann.

Der Gerätetreibername für den XLON USB Adapter lautet: ‚\\.\xlonusb0'

Dieser Name kann über einen Alias auch aus der Registrierung ausgelesen werden. Soll das zu verwendende Netzwerkinterface ausgewählt werden können, so empfiehlt es sich, den

Zugriff über einen Alias-Namen aus der Registrierung auswählbar zu machen. Hierzu können die, unter dem Registrierungsschlüssel

<HKEY_LOCAL_MACHINE\SOFTWARE\LonWorks\DeviceDrivers\>

angezeigten Geräte zur Auswahl angeboten werden.

Den XLON® USB Adapter findet man unter dem Alias-Namen: XLON USB x

x: XLON® USB Netzwerk-Interface-Nummer

z.B. XLON USB 0 => erster im System eingebauter XLON® USB Adapter.

Notwendige Kommando- und Typdefinitionen:

Kommandos unter Windows98/ME:

```
#define LDV_Acquire                  1     // Belegung des Treibers kennzeichnen
#define LDV_Release                  2     // Belegung des Treiber aufheben
#define LDV_Register_Event_Handle    7     // Event-Handle zur Kommunikation anfordern
#define LDV_Read                     10    // Lesen vom Treiber
#define LDV_Write                    11    // Schreiben vom Treiber
```

Typdefinition des Anwendungspuffers:

```
#define MAXLONMSG                    253   // Maximale Länge für Daten im Messagepaket
typedef struct APILNI_Message_Struct
{
        BYTE NiCmd;                        // Network Interface Command
        BYTE Length;                       // Size of ExpAppBuffer
        BYTE ExpAppBuffer[MAXLONMSG];      // Buffer for Data
} APILNI_Message;
```

Struktur des Anwendungspuffers (API LNI Message)

<table>
<tr><td rowspan="2"></td><td>Command</td><td rowspan="2">2 Byte</td></tr>
<tr><td>Length</td></tr>
<tr><td rowspan="3">Length</td><td>Begin of Data</td><td>3 Byte</td></tr>
<tr><td>Network Adress</td><td>11 Byte</td></tr>
<tr><td>Data</td><td>variable Length</td></tr>
</table>

Definition eines Zugriffhandles

```
1.)     HANDLE* phandle = new HANDLE;      // Handle to access the device driver
```

Definition der Anwendungs Puffer

```
2.)     APILNI_Message* ni_in_msg = new APILNI_Message;     // NSI Message In structure
        APILNI_Message* ni_out_msg = new APILNI_Message;    // NSI Message Out structure
```

Windows98/ME:

Die Funktion ‚CreateFile' öffnet den Gerätetreiber für den XLON USB Adapter und liefert ein Handle für den Zugriff auf den Gerätetreiber zurück. Als Gerätetreibername muss ‚\\.\xlonusb0' verwendet werden.

2.) *pHandle = CreateFile("\\\\.\\xlonusb0", GENERIC_READ | GENERIC_WRITE, 0, 0, OPEN_EXISTING, 0, 0);

Mögliche vom Treiber zurückgegebene Fehlercodes:

Fehler:	Zu wenig Speicher für die Allokierung der Treiberpuffer
Fehlercode:	ERROR_NOT_ENOUGH_MEMORY

Nachdem der Treiber geöffnet wurde muss unter Windows98/ME Betriebssystemen der Befehl 'LDV_Aquire' ausgeführt werden. Hierdurch wird dem Gerätetreiber angezeigt, dass auf den Treiber zugegriffen wurde.

3.) DeviceIoControl(*pHandle, MAKELONG(LDV_Acquire, 0), &inBuf, sizeof(char), &RetInfo, sizeof(RetInfo), &nBytesReturned, NULL);

Mögliche vom Treiber zurückgegebene Fehlercodes:

Fehler:	keine

Windows2000/XP:

Die Funktion ‚CreateFile' öffnet den Gerätetreiber für den XLON USB Adapter und liefert ein Handle für den Zugriff auf den Gerätetreiber zurück. Als Gerätetreibername muss ‚\\.\xlonusb0' verwendet werden.

2.) *pHandle = CreateFile("\\\\.\\xlonusb0", GENERIC_READ | GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE, (LPSECURITY_ATTRIBUTES) NULL, OPEN_EXISTING, 0, (HANDLE) NULL);

Mögliche vom Treiber zurückgegebene Fehlercodes:

Fehler:	Zu wenig Speicher für Allokierung Treiberpuffer
Fehlercode:	ERROR_NOT_ENOUGH_MEMORY

6.1.3.2 Registrieren eines Event-Handles

Zur Kommunikation zwischen Treiber und Applikation kann ein gemeinsamer Event-Handle registriert werden. Dazu muß die Applikation einen Handle beim Betriebssystem anfordern. Diesen Handle übergibt die Applikation anschließend dem Treiber. Der Treiber kreiert immer dann einen Event, wenn der Treiber Daten für die Hostapplikation hat.

Windows98/ME:

Kreieren eines Synchronisations-Event-Handles:
1.) CreateCommonEvent(&hEventR3, &hEventR0, FALSE, FALSE);

Übergeben des Event-Handles an den Treiber:
2.) DeviceIoControl(pHandle, MAKELONG(LDV_Register_Event_Handle,0), hEventR0, sizeof(HANDLE), &RetInfo, sizeof(RetInfo), &nBytesReturned, NULL));

Windows2000/XP:

Hier kommt ein anderer Event Mechanismus zum Einsatz. Der Zugriff erfolgt über Overlapped IO, das registrieren eines Event Handle ist somit nicht notwendig.



6.1.3.3 Lesen von Daten vom Gerätetreiber

Zum Lesen von Daten wird unter Windows98/ME das Kommando ‚LDV_Read' in der Windows API Funktion ‚DeviceIoControl' verwendet.

Unter Windows 2000/XP wird die Windows API Funktion ‚ReadFile' verwendet.

Windows98/ME:

```
DeviceIoControl(pHandle, MAKELONG(LDV_Read, 0), NULL, sizeof(char), ni_in_Msg, length,
&nBytesReturned, NULL) ;
```

Mögliche vom Treiber zurückgegebene Fehlercodes:

Fehler:	Vor dem Zugriff auf den Treiber wurde kein LDV_ACQUIRE ausgeführt
Fehlercode:	ERROR_ACCESS_DENIED

Windows2000/XP:

```
ReadFile( pHandle, ni_in_Msg, length+1, &nBytesReturned, NULL );
```

Mögliche vom Treiber zurückgegebene Fehlercodes:

Fehler:	Overlapped IO und keine Daten vorhanden.
Fehlercode:	STATUS_PENDING
Fehler:	Non Overlapped IO und keine Daten vorhanden.
Fehlercode:	STATUS_UNSUCCESSFUL

6.1.3.4 Schreiben von Daten auf den Gerätetreiber

Zum Schreiben von Daten wird unter Windows98/ME das Kommando ‚LDV_Write' in der Windows API Funktion ‚DeviceIoControl' verwendet. Unter Windows 2000/XP wird die Windows API Funktion ‚WriteFile' verwendet.

Windows98/ME:

```
DeviceIoControl(pHandle, MAKELONG(LDV_Write, 0) , ni_out_Msg, length, NULL, sizeof(char),
&nBytesReturned, NULL);
```

Mögliche vom Treiber zurückgegebene Fehlercodes:

Fehler:	Vor dem Zugriff auf den Treiber wurde kein LDV_ACQUIRE ausgeführt
Fehlercode:	ERROR_ACCESS_DENIED
Fehler:	Kein freier Applikation Puffer im Treiber vorhanden
Fehlercode:	ERROR_NOT_ENOUGH_MEMORY
Fehler:	Von der Applikation gesendeter Datensatz zu groß
Fehlercode:	ERROR_ACCESS_DENIED

Windows2000/XP:

```
WriteFile( pHandle, ni_out_Msg, length, &nBytesReturned, NULL );
```

Mögliche vom Treiber zurückgegebene Fehlercodes:

Fehler	Applikation Puffer für schreiben erschöpft.
Fehlercode:	ERROR_NOT_ENOUGH_MEMORY



6.1.3.5 Schließen des Gerätetreibers

Wird die Anwendung beendet, so muss der Treiber geschlossen werden.

Windows98/ME:

Bevor der Treiber endgültig geschlossen werden kann, muss zuerst der Befehl ‚LDV_Release' aufgerufen werden. Hierdurch wird die Belegung des Gerätetreibers aufgehoben.

1.) DeviceIoControl(pHandle, MAKELONG(LDV_Release, 0), &inBuf, sizeof(char), &RetInfo, sizeof(RetInfo), &nBytesReturned, NULL);

Anschließend muss der Treiber geschlossen werden.
2.) CloseHandle(pHandle);

Windows2000/XP:

2.) CloseHandle(pHandle);

6.1.3.6 Wichtige Programmierinformationen

Wird eine eigene Anwendung für den XLON® USB Adapter programmiert, so muss bei der Initialisierung des Gerätes eine Programm-ID in das Gerät programmiert werden. Die zu verwendende Programm ID kann von der Anwendung bestimmt werden.

Die Netzwerk- und Anwendungs-Puffer des Neuron® Prozessors können verändert werden. Hierbei ist zu beachten, dass die maximal zulässige Byte Anzahl für alle verwendeten Puffer 4556 Byte nicht überschreiten darf. Zulässige Puffer Einstellungen sollten mit dem LNS Plug-In ‚Network Interface Buffer Configuration' ermittelt werden (vgl. Kapitel 6.1.2).

☞ In manchen Fällen kann bei falschen Werten der Puffergröße der XLON® USB Adapter funktionslos werden. Dieser Zustand kann nur noch mittels 'Reboot' des Gerätes oder in vereinzelten Fällen gar nicht mehr rückgängig gemacht werden.

Sollte bei der RS485 Variante eine spezielle Übertragungsrate, die in der Eigenschaften Seite (vgl. Kapitel 4.1.3) nicht einstellbar ist, benötigt werden, so muss in der Eigenschaften Seite die Transceiver ID ‚Custom Transceiver' parametriert werden. Dabei ist zu beachten, dass dann die Anwendung dafür verantwortlich ist, die richtigen Einstellungen für den Transceiver und der Übertragungsrate vorzunehmen.

6.2 Applikationsschnittstelle unter Windows CE 3.0

Zum Erstellen einer C/C++ LON Hostapplikation unter Windows CE 3.0 kann prinzipiell die gleiche Dokumentation, wie unter 6.1.3 angegeben, verwendet werden. Allerdings unterscheidet sich das Application Programming Interface (API) zum Zugriff auf den Gerätetreiber unter Windows CE 3.0 von Desktop Windows Betriebssystemen.

Unter Windows CE 3.0 stehen zum Zugriff aus einer eigenen C/C++ LON Hostapplikationen auf den Gerätetreiber die folgenden Standard-Betriebssystemaufrufe (Windows CE 3.0 API) für „Stream Interface Devices" zur Verfügung:

CreateFile()	ReadFile()
CloseHandle()	DeviceIoControl()
WriteFile()	GetLastError()

In den folgenden Unterkapiteln wird ein Überblick über die einzelnen API-Funktionen gegeben. Für eine ausführliche Beschreibung wird auf die Windows CE 3.0 Dokumentation verwiesen, die in der Microsoft MSDN Library bzw. in der Microsoft Windows CE Platform Builder 3.0 Library zu finden ist.

6.2.1 CreateFile()

Prototyp:

```
HANDLE CreateFile(  LPCTSTR lpFileName,
                    DWORD dwDesiredAccess,
                    DWORD dwShareMode,
                    LPSECURITY_ATTRIBUTES lpSecurityAttributes,
                    DWORD dwCreationDisposition,
                    DWORD dwFlagsAndAttributes,
                    HANDLE hTemplateFile );
```

Beschreibung:

Diese Funktion öffnet den durch „lpFileName" spezifizierten Gerätetreiber, wobei sich der Gerätetreibername aus dem Gerätepräfix und dem Geräteindex zusammensetzt, gefolgt von einem Doppelpunkt, z.B. "LON1: ", "LON2: ", "LON3: ", usw. Der Aufbau des Gerätenamens wurde bereits in Kapitel 3.3.2.1 ausführlich dargestellt. Ausführliche Informationen zu den weiteren Funktionsparametern und zur Funktion CreateFile() im Allgemeinen sind der Windows CE 3.0 Dokumentation zu entnehmen.

Beispiel:

```
HANDLE myHandle;

myHandle = CreateFile(TEXT("LON1:"),
                    GENERIC_READ | GENERIC_WRITE,
                    0x00, NULL
                    OPEN_EXISTING,
                    FILE_ATTRIBUTE_NORMAL,
                    NULL )
```

Rückgabewert:

Konnte der Gerätetreiber erfolgreich geöffnet werden, wird ein Handle auf den Gerätetreiber zurückgegeben. Über diesen Handle können weitere Operationen auf den Gerätetreiber durchgeführt bzw. der Gerätetreiber wieder geschlossen werden. Im Fehlerfall wird

INVALID_HANDLE_VALUE zurückgeliefert, dann können detailliertere Informationen zur Fehlerursache über den Aufruf der Funktion GetLastError() erlangt werden.

Fehlerursachen:

Als wahrscheinliche Fehlerursachen für ein Fehlschlagen dieser Funktion kommen ein nicht geladener Gerätetreiber, ein falscher Gerätetreibername, ein vorherig nicht korrekt geschlossener Gerätetreiber oder fehlerhaft gesetzte Funktionsparameter in Frage.

6.2.2 CloseHandle()

Prototyp:

BOOL **CloseHandle**(HANDLE hObject);

Beschreibung:

Diese Funktion schließt den über den Handle „hObject" spezifizierten Gerätetreiber. Als Funktionsparameter ist der bei erfolgreichem Aufruf der Funktion CreateFile() zurückgelieferte Handle auf den Gerätetreiber zu übergeben. Ausführlichere Informationen zur Funktion CloseHandle() sind der Windows CE 3.0 Dokumentation zu entnehmen.

Beispiel:

BOOL bResult;

bResult = **CloseHandle**(myHandle);

Rückgabewert:

Konnte der Gerätetreiber erfolgreich geschlossen werden, wird der Wert „TRUE" zurückgegeben, ansonsten „FALSE". In letzterem Fall können detailliertere Informationen zur Fehlerursache über den Aufruf der Funktion GetLastError() erlangt werden. Nach einem erfolgreichen Aufruf von CloseHandle() ist der übergebene Handle nicht mehr gültig und kann nicht mehr für Operationen auf den Gerätetreiber verwendet werden.

Fehlerursachen:

Als wahrscheinliche Fehlerursache für ein Fehlschlagen dieser Funktion kommt ein ungültiger Handle auf den Gerätetreiber in Frage.

6.2.3 WriteFile()

Prototyp:

BOOL **WriteFile**(HANDLE *hFile*,
LPCVOID *lpBuffer*,
DWORD *nNumberOfBytesToWrite*,
LPDWORD *lpNumberOfBytesWritten*,
LPOVERLAPPED *lpOverlapped*);

Beschreibung:

Mit dieser Funktion werden Daten von einer LON Applikation auf den Gerätetreiber und somit auf das XLON®-Netzwerkinterface geschrieben. Aufrufe dieser Funktion geschehen asynchron, d.h. die Funktion kehrt sofort zurück, sobald die Daten in den internen Puffer des Gerätetreibers übernommen wurden oder dabei ein Fehler aufgetreten ist. Die Verarbeitung der Daten verläuft dann parallel zur LON Applikation im Hintergrund.

Das jeweilige XLON®-Netzwerkinterface wird über seinen Handle „hFile" spezifiziert. Der

Zeiger „lpBuffer“ muss auf eine Datenstruktur vom Typ APILNI_Message zeigen, die auch als Application Layer Buffer bezeichnet wird. Die Größe dieser Datenstruktur ist variabel, der Aufbau ist im folgenden Punkt dargestellt. Über den Parameter „nNumberOfBytesToWrite“ wird die für den jeweiligen Aufruf gültige Größe dieser variablen Datenstruktur festgelegt. Mittels des Parameters „lpNumberOfBytesWritten“ wird die tatsächlich an das Netzwerkinterface übertragene Anzahl von Bytes zurückgeliefert. Im Erfolgsfall sind diese beiden Werte identisch und ungleich Null. Der Parameter „lpOverlapped“ hat unter Windows CE 3.0 keine Verwendung. Ausführliche Informationen zu den einzelnen Funktionsparametern und zur Funktion WriteFile() im Allgemeinen sind der Windows CE 3.0 Dokumentation zu entnehmen.

Application Layer Buffer:

Der folgende C-Code definiert den Datentyp „APILNI_Message“ für Application Layer Buffer, wie im Echelon NSI Firmware User's Guide spezifiziert. Der Aufbau des Strukturelements „ExpAppBuffer[]“ wird im „LonWorks Host Application Programmer's Guide“ ausführlich erläutert.

```
#define MAXLONMSG 253

typedef struct APILNI_Message_Struct {
        BYTE NiCmd;                        // NSI command
        BYTE Length;                       // Size of ExpAppBuffer
        BYTE ExpAppBuffer[MAXLONMSG];      // message data
} APILNI_Message;
```

Die folgende Tabelle zeigt nochmals den Aufbau des Application Layer Buffers in strukturierter Form.

command	2 Bytes	Application Layer Header
length		
message header	3 Bytes	ExpAppBuffer[MAXLONMSG], Größe max. 253 Bytes
network address	11 Bytes	
message data	variable Länge	

Beispiel:

```
BOOL bResult;
DWORD dwBytesWritten;
APILNI_Message lni_msg = { niRESET, 0x00 };       // Puffer mit Reset Kommando an NSI

bResult = WriteFile(    myHandle,
                        (LPCVOID)&lni_msg,
                        0x02,
                        &dwBytesWritten,
                        NULL );
```

Rückgabewert:

Konnte die Schreiboperation auf den Gerätetreiber erfolgreich durchgeführt werden, wird der Wert „TRUE“ zurückgegeben, ansonsten „FALSE“. In letzterem Fall können detailliertere Informationen zur Fehlerursache über den Aufruf der Funktion GetLastError() erlangt werden.

Fehlerursachen:

Als wahrscheinliche Fehlerursachen für ein Fehlschlagen dieser Funktion kommen ein ungültiger Handle auf den Gerätetreiber, ein fehlerhaft aufgebauter Application Layer Buffer oder ein außerhalb des gültigen Bereichs liegender bzw. nicht mit der tatsächlichen Länge des Application Layer Buffer übereinstimmender Parameter „nNumberOfBytesToWrite" in Frage.

6.2.4 ReadFile()

Prototyp:

BOOL **ReadFile**(HANDLE *hFile*,
LPVOID *lpBuffer*,
DWORD *nNumberOfBytesToRead*,
LPDWORD *lpNumberOfBytesRead*,
LPOVERLAPPED *lpOverlapped*);

Beschreibung:

Mit dieser Funktion werden Daten durch eine LON Applikation vom Gerätetreiber und somit vom XLON®-Netzwerkinterface gelesen. Aufrufe dieser Funktion geschehen asynchron, d.h. die Funktion kehrt sofort zurück, sobald die Daten vom internen Puffer des Gerätetreibers übernommen wurden. Falls keine Daten zur Verfügung stehen oder ein Fehler aufgetreten ist, kehrt diese Funktion ebenfalls sofort zurück, d.h. es wird nicht auf das Eintreffen von Daten vom XLON®-Netzwerkinterface gewartet.

Das jeweilige XLON®-Netzwerkinterface wird über seinen Handle „hFile" spezifiziert. Der Zeiger „lpBuffer" muss auf eine Datenstruktur vom Typ APILNI_Message zeigen, die auch als Application Layer Buffer bezeichnet wird. Die Größe dieser Datenstruktur ist variabel, sollte jedoch bei Leseoperationen auf den Maximalwert gesetzt werden, da die Anzahl der tatsächlich zu lesenden Daten beim Aufruf der Funktion noch nicht bekannt ist. Der Aufbau ist im folgenden Punkt dargestellt. Über den Parameter „nNumberOfBytesToRead" wird die für den jeweiligen Aufruf gültige Größe dieser variablen Datenstruktur festgelegt. Mittels des Parameters „lpNumberOfBytesRead" wird die tatsächlich vom Netzwerkinterface gelesene Anzahl von Bytes zurückgeliefert. Im Erfolgsfall ist dieser Wert kleiner oder gleich „nNumberOfBytesToRead" und ungleich Null. Der Parameter „lpOverlapped" hat unter Windows CE 3.0 keine Verwendung. Ausführliche Informationen zu den einzelnen Funktionsparametern und zur Funktion ReadFile() im Allgemeinen sind der Windows CE 3.0 Dokumentation zu entnehmen.

Application Layer Buffer:

Der folgende C-Code definiert den Datentyp „APILNI_Message" für Application Layer Buffer, wie im Echelon NSI Firmware User's Guide spezifiziert. Der Aufbau des Strukturelements „ExpAppBuffer[]" wird im „LonWorks Host Application Programmer's Guide" ausführlich erläutert.

```
#define MAXLONMSG 253

typedef struct APILNI_Message_Struct {
        BYTE NiCmd;                      // NSI command
        BYTE Length;                     // size of ExpAppBuffer
        BYTE ExpAppBuffer[MAXLONMSG];    // message data
} APILNI_Message;
```

Die folgende Tabelle zeigt nochmals den Aufbau des Application Layer Buffer in strukturierter Form.

<table>
<tr><td>command</td><td rowspan="2">2 Bytes</td><td rowspan="2">Application Layer Header</td></tr>
<tr><td>length</td></tr>
<tr><td>message header</td><td>3 Bytes</td><td rowspan="3">ExpAppBuffer[MAXLONMSG],
Größe max. 253 Bytes</td></tr>
<tr><td>network address</td><td>11 Bytes</td></tr>
<tr><td>message data</td><td>variable Länge</td></tr>
</table>

Beispiel:

```
BOOL bResult;
DWORD dwBytesRead;
APILNI_Message lni_msg;              // Application Layer Buffer für Message vom NSI

bResult = ReadFile( myHandle,
                    (LPCVOID)&lni_msg,
                    255,
                    &dwBytesRead,
                    NULL );
```

Rückgabewert:

Konnte die Leseoperation auf den Gerätetreiber erfolgreich durchgeführt werden, wird der Wert „TRUE" zurückgegeben. Ist die Anzahl der gelesenen Bytes, die in „lpNumberOfBytesRead" zurückgegeben wird gleich Null, so waren keine Daten verfügbar. Ist die Leseoperation auf den Gerätetreiber fehlgeschlagen, so wird der Wert „FALSE" zurückgegeben. In diesem Fall können detailliertere Informationen zur Fehlerursache über den Aufruf der Funktion GetLastError() erlangt werden.

Fehlerursachen:

Als wahrscheinliche Fehlerursachen für ein Fehlschlagen dieser Funktion kommen ein ungültiger Handle auf den Gerätetreiber oder ein außerhalb des gültigen Bereichs liegender bzw. ein nicht mit der benötigten Länge des Application Layer Buffers übereinstimmender Parameter „nNumberOfBytesToRead" in Frage.

6.2.5 DeviceIoControl()

Prototyp:

BOOL **DeviceIoControl**(HANDLE *hFile*,
DWORD *dwIoControlCode*
LPVOID *lpInBuffer*,
DWORD *nInBufferSize*,
LPVOID *lpOutBuffer*,
DWORD *nOutBufferSize*,
LPDWORD *lpBytesReturned*,
LPOVERLAPPED *lpOverlapped*);

Beschreibung:

Mit dieser Funktion können bestimmte Operationen auf den Gerätetreiber durchgeführt werden, die mit den bislang beschriebenen Funktionen nicht möglich sind. Zur Zeit sind dies die Operationen „GetVersion" und „ReadWait", die durch die Kommandos „IOCTL_XLON_GETVERSION" bzw. „IOCTL_XLON_READWAIT" aufgerufen werden können.

Die Operation „GetVersion“ ermöglicht das Auslesen einer Version aus dem Gerätetreiber. Die Operation „ReadWait“ ist die synchrone (blockierende) Variante der Funktion ReadFile(), d.h. die Anwendung wartet hier bis ein Rückgabewert vorliegt. Diese beiden mittels DeviceIoControl() durchführbaren Operation werden im folgenden genauer Erläutert.

6.2.5.1 „GetVersion“ über DeviceIoControl()

Beschreibung:

Die Operation „GetVersion“ ist über die API-Funktion DeviceIoControl() realisiert und ermöglicht das Auslesen einer Versionskennung aus dem Gerätetreiber. Der I/O-Control-Code für diese Operation ist als „IOCTL_XLON_GETVERSION“ definiert. Die Treiberversion ist in Form eines DWORD kodiert, wobei jedes der 4 Bytes binär kodiert ist und für eine dezimale Ziffer steht. Die Major-Version ist in Bit 16 bis Bit 23 (Byte 2) und die Minor-Version in Bit 8 bis Bit 15 (Byte 1) kodiert, die restlichen Bits (Byte 0 und Byte 3) haben momentan keine Bedeutung. Ein ausgelesenes DWORD von 0x00010200 entspricht somit dem Wert 0.1.2.0, d.h. die Treiberversion lautet 1.2.

Beim Aufruf von DeviceIoControl() ist das jeweilige XLON®-Netzwerkinterface über seinen Handle „hFile“ zu spezifizieren. Der Parameter „dwIoControlCode“ ist mit dem Kommando „IOCTL_XLON_GETVERSION“ zu besetzen. Die Parameter „lpInBuffer“ bzw. „nInBufferSize“ werden nicht benötigt. Für den Parameter „lpOutBuffer“ wird ein Zeiger auf ein DWORD übergeben, in diesem wird später die Treiberversion abgelegt. In „nOutBufferSize“ wird die Größe dieses DWORD in Byte übergeben. Im Parameter „lpBytesReturned“ wird die Anzahl der gelesenen Bytes zurückgeliefert. Der Parameter „lpOverlapped“ hat unter Windows CE 3.0 keine Verwendung. Ausführliche Informationen zu den einzelnen Funktionsparametern und zur Funktion DeviceIoControl() im Allgemeinen sind der Windows CE 3.0 Dokumentation zu entnehmen.

Beispiel:

```
#define IOCTL_XLON_GETVERSION (DWORD)0x01 // IOCTL-Code für Kommando
                                                        „GetVersion“
BOOL bResult;
DWORD dwVersion, dwBytesReturned;

bResult = DeviceIoControl(    myHandle,
                              IOCTL_XLON_GETVERSION,
                              NULL, 0,
                              &dwVersion,
                              sizeof( dwVersion ),
                              &dwBytesReturned
                              NULL );
```

Rückgabewert:

Konnte die Treiberversion erfolgreich aus dem Gerätetreiber ausgelesen werden, wird der Wert „TRUE“ zurückgegeben, andernfalls wird der Wert „FALSE“ zurückgegeben. In diesem Fall können detaillierte Informationen zur Fehlerursache über den Aufruf der Funktion GetLastError() erlangt werden.

Fehlerursachen:

Als wahrscheinliche Fehlerursache für ein Fehlschlagen dieser Funktion kommt ein ungültiger Handle auf den Gerätetreiber in Frage.

6.2.5.2 „ReadWait" über DeviceIoControl()

Beschreibung:

Die Operation „ReadWait" ist die synchrone (blockierende) Variante der Funktion ReadFile(), d.h. die Anwendung wartet hier bis ein Rückgabewert vorliegt. Mit dieser Operation werden Daten durch eine LON Applikation vom Gerätetreiber und somit vom XLON®-Netzwerkinterface gelesen. Sind keine Daten im internen Puffer des Gerätetreibers vorhanden, wird eine frei definierbare Zeit auf das Eintreffen von Daten gewartet, bevor der Aufruf zurückkehrt (Blocking Call). Wird eine Wartezeit von „Null" festgelegt, ist die funktionsweise identisch zum Aufruf der Funktion ReadFile(), wird eine Wartezeit von „INFINITE" festgelegt, wird ohne Timeout gewartet.

Beim Aufruf von DeviceIoControl() ist das jeweilige XLON®-Netzwerkinterface über seinen Handle „hFile" zu spezifizieren. Der Parameter „dwIoControlCode" ist mit dem Kommando „IOCTL_XLON_READWAIT" zu besetzen. Im Parameter „lpInBuffer" wird die Wartezeit für die Operation „ReadWait" übergeben, wobei es sich um einen Zeiger auf ein DWORD handeln muss. Der Wert dieses DWORD spezifiziert die Wartezeit in Millisekunden, bei einem Wert von „INFINITE" wird solange gewartet, bis Daten vom XLON®-Netzwerkinterface eingetroffen sind oder der Treiber geschlossen wird. Der Parameter „nInBufferSize" enthält die Länge des vorigen DWORD mit der Wartezeit. Der Zeiger „lpOutBuffer" muss auf eine Datenstruktur vom Typ APILNI_Message zeigen, die auch als Application Layer Buffer bezeichnet wird. Die Größe dieser Datenstruktur ist variabel, sollte jedoch bei Leseoperationen auf den Maximalwert gesetzt werden, da die Anzahl der tatsächlich zu lesenden Daten beim Aufruf der Funktion noch nicht bekannt ist. Der Aufbau ist im folgenden Punkt dargestellt. Über den Parameter „nOutBufferSize" wird die für den jeweiligen Aufruf gültige Größe dieser variablen Datenstruktur festgelegt. Mittels des Parameters „lpBytesReturned" wird die tatsächlich vom Netzwerkinterface gelesene Anzahl von Bytes zurückgeliefert. Im Erfolgsfall ist dieser Wert kleiner oder gleich „nOutBufferSize" und ungleich Null. Der Parameter „lpOverlapped" hat unter Windows CE 3.0 keine Verwendung. Ausführliche Informationen zu den einzelnen Funktionsparametern und zur Funktion ReadFile() im Allgemeinen sind der Windows CE 3.0 Dokumentation zu entnehmen.

Application Layer Buffer:

Der folgende C-Code definiert den Datentyp „APILNI_Message" für Application Layer Buffer, wie im Echelon NSI Firmware User's Guide spezifiziert. Der Aufbau des Strukturelements „ExpAppBuffer[]" wird im „LonWorks Host Application Programmer's Guide" ausführlich erläutert.

```
#define MAXLONMSG 253

typedef struct APILNI_Message_Struct {
        BYTE NiCmd;                          // NSI command
        BYTE Length;                         // Size of ExpAppBuffer
        BYTE ExpAppBuffer[MAXLONMSG];        // Message data
} APILNI_Message;
```

Die folgende Tabelle zeigt nochmals den Aufbau des Application Layer Buffers in grafischer Form.

<table>
<tr><td>command</td><td rowspan="2">2 Bytes</td><td rowspan="2">Application Layer Header</td></tr>
<tr><td>length</td></tr>
</table>



message header	3 Bytes	ExpAppBuffer[MAXLONMSG], Größe max. 253 Bytes
network address	11 Bytes	
message data	variable Länge	

Beispiel:

```
#define IOCTL_XLON_READWAIT (DWORD)0x00 // IOCTL-Code für Kommando
                                                          „ReadWait“
BOOL bResult;
DWORD dwTimeout = INFINITE;
DWORD dwBytesReturned;
APILNI_Message lni_msg                // Application Layer Buffer für Message vom NSI

bResult = DeviceIoControl(   myHandle,
                             IOCTL_XLON_READWAIT,
                             &dwTimeout,
                             sizeof( dwTimeout ),
                             (LPVOID)&lni_msg,
                             255,
                             &dwBytesReturned
                             NULL );
```

Rückgabewert:

Konnte die Leseoperation auf den Gerätetreiber erfolgreich durchgeführt werden, wird der Wert „TRUE“ zurückgegeben. Ist die Anzahl der gelesenen Bytes, die in „lpBytesReturned“, zurückgegeben wird, gleich Null, so waren nach Ablauf der Wartezeit immer noch keine Daten verfügbar. Ist die Leseoperation auf den Gerätetreiber fehlgeschlagen, so wird der Wert „FALSE“ zurückgegeben. In diesem Fall können detailliertere Informationen zur Fehlerursache über den Aufruf der Funktion GetLastError() erlangt werden.

Fehlerursachen:

Als wahrscheinliche Fehlerursachen für ein Fehlschlagen dieser Funktion kommen ein ungültiger Handle auf den Gerätetreiber oder ein außerhalb des gültigen Bereichs liegender bzw. ein nicht mit der benötigten Länge des Application Layer Buffers übereinstimmender Parameter „nOutBufferSize“ in Frage.

6.2.6 GetLastError()

Prototyp:

DWORD **GetLastError**(<u>void</u>);

Beschreibung:

Diese Funktion liefert detailliertere Informationen zur Fehlerursache, wenn eine der Funktionen CreateFile(), CloseHandle(), ReadFile(), WriteFile() bzw. DeviceIoControl() fehlschlägt. Ausführlichere Informationen zur Funktion GetLastError() sind der Windows CE 3.0 Dokumentation zu entnehmen.

Beispiel:

```
DWORD dwLastError;

dwLastError = GetLastError();
```

Rückgabewert:

Fehlercode der letzten Operation. Für die XLON Gerätetreiber sind unter Windows CE 3.0 die folgenden Fehlercodes definiert:

```
typedef enum
{
        LONDEV_SUCCESS = 0,             // no error detected
        LONDEV_NOT_FOUND,               // device not found
        LONDEV_ALREADY_OPEN,            // device already open
        LONDEV_NOT_OPEN,                // not a handle to a open device
        LONDEV_DEVICE_ERR,              // device detect error
        LONDEV_INVALID_DEVICE_ID,       // invalid device id was detected
        LONDEV_NO_MSG_AVAIL,            // no message available
        LONDEV_NO_BUFF_AVAIL,           // buffer is full
        LONDEV_NO_RESOURCES,            // no more resources available
        LONDEV_INVALID_BUF_LEN,         // invalid buffer length
        LONDEV_DEVICE_BUSY              // device is busy
} LonDevCode;
```

6.3 Applikationsschnittstelle unter Linux

Derzeit ist für den XLON® USB Adapter kein Linux Treiber verfügbar.

7 Anhang

7.1 EG-Konformitätserklärung

Das Produkt

Fabrikat	XLON USB
Typbezeichnung(en)	USB4-WM-FTT, USB4-RJ-485 und 229838 Interface LON-PC USB
	ist entwickelt, konstruiert und gefertigt in Übereinstimmung mit den EG-Richtlinien 89/336/EG, geändert 92/31/EG in alleiniger Verantwortung von
Firma	DH electronics GmbH Am Anger 8 83346 Bergen Germany

Folgende harmonisierten Fachgrundnormen sind angewandt:

EMV Störaussendung
Fachgrundnorm: EN 50081-1

EMV Störfestigkeit
Fachgrundnorm: EN 50082-1
Darüber hinaus wurde die EN 50082-2 für industrielle Anforderungen angewandt.

Folgende nationalen Normen, Richtlinien und Spezifikationen sind zusätzlich angewandt:

Keine.

Eine Technische Dokumentation ist vollständig vorhanden. Die zum Produkt gehörende Betriebsanleitung liegt in der Originalfassung vor.

Bergen, den 17. Mai 2001
Ort, Datum

Dipl.-Ing. (FH) Stefan Daxenberger
Geschäftsleitung

Diese EG-Konformitäts-Erklärung gilt für Seriengeräte und ist daher als Kopie gültig.



8 Änderungsstand Dokument

Version	Ausgabe-datum	Änderung	Status	Veran-lasser	Kommentar
1.0	21.02.03	Grundversion	verab-schiedet	StS	